



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 1/81  
Filename: ASTCDetailedDesign

<b>TITLE</b>	ASTC DETAILED DESIGN
<b>DOCUMENT TYPE</b>	REPORT
<b>DOC No.</b>	AMST/ STCD /1/A
<b>ISSUE No.</b>	1
<b>DATE</b>	Oct 06

Prepared by: A.Bucconi (CARSO)

Approved by: P.Trampus (CARSO)



## AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 2/81  
Filename: ASTCDetailedDesign

### CHANGE RECORD

Issue	Date	Affected Pages	Description of Change	Change Authority



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 3/81  
Filename: *ASTCDetailedDesign*

### Table of Contents

1.	CIRCUIT DESCRIPTION .....	4
1.1.	CCD Chip .....	4
1.2.	Front-End Electronics .....	5
2.	CONNECTORS.....	9
3.	BILL OF MATERIALS.....	12
4.	VHDL .....	16
4.1.1.	ccd_driver.vhd .....	23
4.1.2.	gen_buff.vhd .....	25
4.1.3.	div.vhd .....	29
4.1.4.	count_ck.vhd .....	30
4.1.5.	my_count15.vhd .....	31
4.1.6.	count_row.vhd .....	37
4.1.7.	ccd_fsm.vhd .....	37
4.1.8.	decod.vhd .....	45
4.1.9.	err_complete.vhd .....	46
4.1.10.	rx_complete.vhd .....	47
4.1.11.	rx_read.vhd .....	48
4.1.12.	rx_ck_rec.vhd .....	49
4.1.13.	my_and7.vhd .....	51
4.1.14.	my_and8.vhd .....	51
4.1.15.	rx_sync_gen.vhd .....	52
4.1.16.	rx_data_rec.vhd .....	53
4.1.17.	my_xor14.vhd .....	55
4.1.18.	hk_complete.vhd .....	57
4.1.19.	tx_complete.vhd .....	59
4.1.20.	tx_mux.vhd .....	60
4.1.21.	my_mux4.vhd .....	62
4.1.22.	tx_count.vhd .....	63
4.1.23.	tx_data_gen.vhd .....	65
4.1.24.	my_xor14.vhd .....	66
4.1.25.	tx_ds_gen.vhd .....	67
5.	SCHEMATICS .....	71

### List of Tables

# 1. CIRCUIT DESCRIPTION

## 1.1. CCD Chip

The camera is based on the E2V type CCD87-00. The CCD, that has been released very recently on the market, matches the traditional MPP technology aimed at reducing the thermal noise with the introduction of a innovative type of amplifier at the read out section of the CCD that allows to lower and even almost cancel the CCD read-out noise (**L3Vision technology**). Main CCD features are pixel size of  $16 \times 16 \mu\text{m}$  and overall sensitive area of 512x512 pixels.

The requirements for CCD biasing and phases control are far more complicated than those of normal E2V CCDs.

The continuous bias voltages may be obtained from a positive voltage of  $V_{DD} = +35\text{V}$  and a negative voltage of  $V_{SS} = -5\text{V}$ . The two voltages may be derived from a switched power supply fed by the external  $+12\text{V}$  voltage. The SPS topology should be chosen so that to optimise the number of components, overall size, power consumption and noise.

The clocks necessary for driving CCD phases also differ from the other standard E2V CCDs. The type CCD87-00 requires four phases for vertical transfer as long as the maximum full well capacity of  $250\text{Ke-}$  is the concern. They can be limited to three (third and fourth shorted) when this requirement is not vital for the application. However in our design because of the lack of antiblooming drain it's preferable to use four phases due to the high dynamic range of the stellar images in order to avoid saturation and consequent blooming of charges spreading across the CCD area.

The block diagram for the power supply and the CCD clock drivers is shown in the following figure.

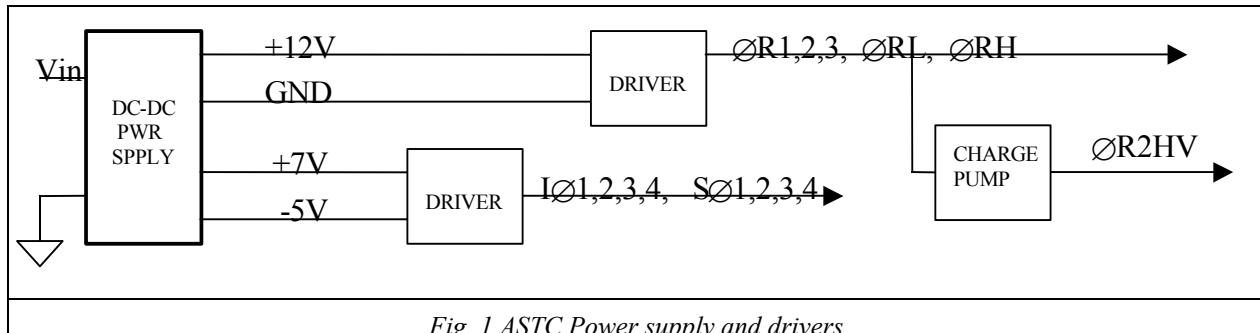


Fig. 1 ASTC Power supply and drivers

The gate  $\mathcal{O}R2HV$  controls the charges multiplication process inside the shift register. The driving signal can be either a sinusoidal or a square wave with level ranging between  $0 \div 50\text{V}$  and it's synchronous with the CCD signal  $\mathcal{O}R2$ . The value of the driving voltage has to be set very accurately as the effective multiplication gain heavily depends on it.

The CCD read-out is done through a procedure of the double correlated sampling (CDS). In the new camera an improved method of clamping the CCD signal will be used. Instead of picking just one sample of the CCD output signal related to the pixel charge, a double sample for each pixel is taken, one during the pixel reset and the other during pixel charge transit. The two samples are then digitally converted and the difference between them is computed. Thus any kind of analogue offsets result ineffective on the pixel charge evaluation process. This digital approach has an important impact of the CCD read-out rate and is made viable through the use of the high speed FPGA placed on board of the camera.

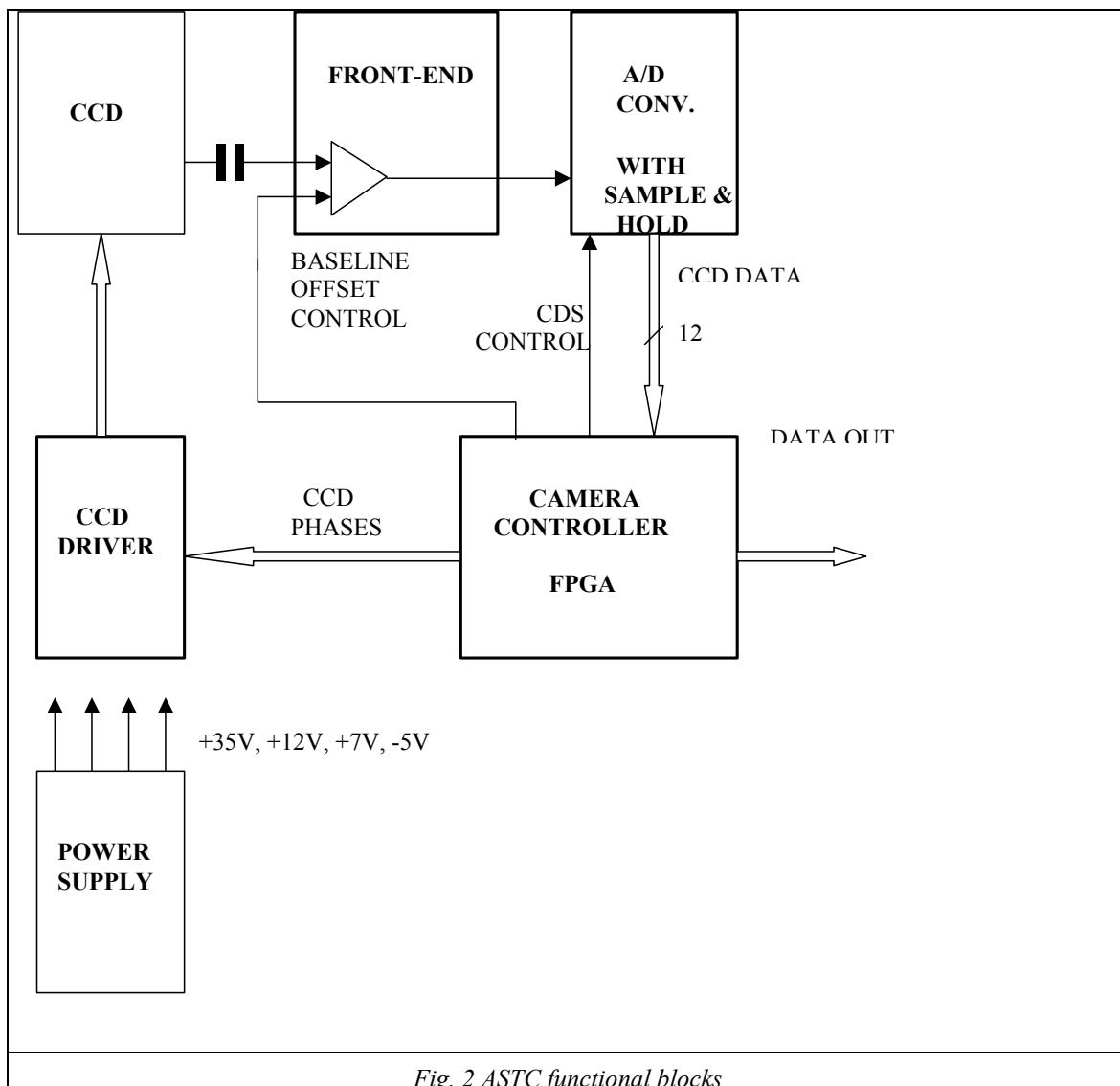
The most important parameters of the camera are summarized in the following table:

*Table 1 CCD main parameters*

Type	E2V CCD87-00
Image	512 x 512 pixels
Image Area Size	8.19 x 8.19 mm
Pixel size	16 x 16 $\mu\text{m}$
Frame Transfer Operation	Yes
CCD operating temperature	-20 $\div$ 35 $^{\circ}\text{C}$
CCD Temperature stability	1 $^{\circ}\text{C}/\text{min}$ (TBD)
CCD temperature monitoring accuracy	1 $^{\circ}\text{C}$

## 1.2. Front-End Electronics

The block diagram of the camera with the double correlated sampling (CDS) is reported below.



### CCD Driver

The task of CCD driver is to provide the necessary current to the CCD gates capacitance. A buffer able to drive a



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 6/81  
Filename: *ASTCDetailedDesign*

capacitance of about 4.1 nF in the voltage range of 0÷12V at the speed of 8 MHz (which is rate of AMICA camera) will be suitable for the CCD needs.

## Preamplifier

The way the preamp is interfaced to the CCD (AC capacitive coupling) poses some requirements to the preamp input impedance, which has to be high enough to avoid voltage decreases on constant signals. The preamp gain can be limited in the range of 1÷10. Because of the CCD internal gain feature the preamp gain may be maintained constant during observations. Its value can be set so that at the minimum CCD gain the stellar magnitude  $M_V=0$  gives rise to an output level that matches the maximum allowed by the A/D converter input range. For example, let it be:

- 0÷1V input voltage range of the A/D converter
- 6 minimum CCD gain
- 108Ke- the maximum pixel charge corresponding to a solar star of magnitude  $M_V=0$
- 1.15  $\mu$ V/e- output LS amplifier responsivity

The signal level at the output of the CCD is then found to be  $V_{out}=.745$  V. In this case a preamp gain of  $G_{AMP}=1.3$  seems to be appropriate.

In the case that the high responsivity amplifier is selected instead of the large signal one, an output level of  $V_{out}=.57$  V is obtained for the same input signal. Thus the value of preamp  $G_{AMP}=1.7$  can be set.

## A/D Converter

The choice of digital clamp requires that the A/D conversion rate doubles compared to that of analogue clamp for the same data throughput. Therefore a speed of 20 MSample/sec for the A/D converter seems to be suitable.

The minimum requested resolution of the A/D converter can be determined by taking into account the noise superimposed to the signal in normal operations. The camera noise, after the reset noise has been cancelled, is formed, in addition to the photonic component here left out, by the thermal and read-out components.

It may be seen that a quantizing level of 12 bit/Sample, which corresponds to a quantum unit of  $1V/4096 = 244 \mu$ V, is appropriate for the application. A better choice would be sampling at 14 bits/sample, what would correspond to a quantum unit of  $1V/16384 = 61 \mu$ V. But technological advances have still to be made in order to make this choice suitable for our application.

## Camera Controller - FPGA

All the control functions of the new camera will be taken over by an FPGA mounted on board. That means great advantage on overall size, power consumption and reliability. The control functions comprise:

- timing generation
- CCD phases generation and control
- read-out and CDS operations control
- A/D data output flow control
- Either control or, optionally, logical implementation of the complete SpaceWire interface.

The read-out procedures of a camera implementing the finder functions are in general straightforward. Due to the need of rather high image acquisition rate, the camera is simply asked for:

- constant acquisition periods of image acquisition at a fixed rate
- sequential and automatic way of image delivery to the acquisition system for their processing

When the implementation is made through these two rules the camera connection to the main computer can be limited to a unidirectional link. A bidirectional connection is necessary if the camera needs some real-time controls under the command of the central computer. Working in hostile environment like the space it may be essential to be able to modify some of the camera parameters in order to optimize the operating conditions during the camera life time. Parameters requiring possible in field settings are:



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 7/81  
Filename: ASTCDetailedDesign

- CCD bias voltages whose optimum values may vary with time due to cumulative radiation effects
- preamp offset voltage to take into account both:
  - dark noise who level is influenced by radiation
  - possible optimising procedures in the automatic star positions evaluation

The description that follows describes the camera control modes starting from the aspects related to an automatic independent acquisition procedure. After that the method to obtain a remote control on the most important parameters is presented.

The principal camera functions are as follows:

- timing generation of the CCD clocks
- control of the CCD clocks
- read-out parameters synchronization
- CDS control and processing
- data flow control to the central computer
- state machine implementation for the whole camera control

The central heart of the CCD controller is formed by a state machine that supervises all the functions necessary for commanding and reading out the CCD. Each elementary CCD operation is associated to one state of the machine.

### ***SpaceWire interface***

SpaceWire interface (ESA standard ECSS-E-50-12) provides a unified infrastructure for connecting together electronic modules in space environment with a high speed serial digital link. It covers the following normative protocols levels:

- Physical level:
- Connectors: 9 pin micro-miniature D-type
- Cables: 4 screened twisted pairs with overall shield
- Cable assemblies
- Signal level:
- Voltage levels: LVDS electrical characteristics
- Data encoding: Data-Strobe encoding
- Signal timing and data rate:
- serial digital links: 2÷400 Mbit/sec
- typical distance: 10metrs
- EMC specifications
- Character level: which defines:
  - data characters used to hold data
  - control characters used to manage the data flow across a link
- Exchange level: which defines:
  - Link initialisation
  - Flow control



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 8/81  
Filename: *ASTCDetailedDesign*

- Detection of Link Errors
- Link error Recovery
- Packet level: defines packet structure

A SpaceWire network is composed by:

- Point-to-point links
- Nodes: source or destination of a packet
- Routers: switch connecting several links that routes packets from one link to another

From the point of view of how the SpaceWire interface will be used inside the AMS project the AMICA camera represents a node that generates information and receives, when necessary, simple commands from the main computer, which represents another node. For this reason from all the specification of the SpaceWire only the part will be taken into account that deals with nodes, packet formation and link initialisation.

*Table 2 ASTC main parameters*

A/D Conversion	12 bit
Readout Noise	22 µV/pixel (3.2 e <sup>-</sup> /pixel) @10°C
Readout Frequency	30 Hz
Integration Time	33 ms
Readout Modes	Transfer from image to store and then to register
Readout Time	33 ms
Readout per Field	1



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 9/81  
Filename: ASTCDetailedDesign

## 2. CONNECTORS

ASTCC 244.300 J2		ASTCE 244.500 J2	
<b>244.640 - PIGTAIL GLENAIR MWDM-2-L-9-P-6-J-7-8-M (24 AWG TEFLON INSULATED WIRES) LENGTH 8"</b>			
PIN	SIGNAL	PIN	SIGNAL
1	+ 8 Vdc	1	+ 8 Vdc
2	GND	2	GND
3	- 8 Vdc	3	- 8 Vdc
4	GND	4	GND
5	+ 27 Vdc	5	+ 27 Vdc
6	GND	6	GND
7	THERMISTOR+	7	THERMISTOR+
8	THERMISTOR-	8	THERMISTOR-
9	HV MONITOR	9	HV MONITOR

ASTCC 244.300 J1	ASTCE 244.500 J1
<b>244.650 - MICRO COAX CABLE w\ SMA PIN CONN. LENGTH 8"</b>	
CCD signal	CCD signal



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 10/81  
Filename: ASTCDetailedDesign

ASTCD 244.400 J1		ASTCE 244.500 J3	
<b>244.630 - PIGTAIL GLENAIR MWDM-2-L-25-P-6-J-7-8-M (24 AWG TEFLON INSULATED WIRES) LENGTH 8"</b>			
PIN	SIGNAL	PIN	SIGNAL
1	$S\emptyset 2$	1	$S\emptyset 2$
2	+ 7 Vdc	2	+ 7 Vdc
3	+ 7 Vdc	3	+ 7 Vdc
4	THERMISTOR+	4	THERMISTOR+
5	GND	5	GND
6	GND	6	GND
7	THERMISTOR-	7	THERMISTOR-
8	- 5 Vdc	8	- 5 Vdc
9	- 5 Vdc	9	- 5 Vdc
10	$\emptyset R$	10	$\emptyset R$
11	+ 12 Vdc	11	+ 12 Vdc
12	+ 12 Vdc	12	+ 12 Vdc
13	$I\emptyset 1$	13	$I\emptyset 1$
14	$S\emptyset 1$	14	$S\emptyset 1$
15	$S\emptyset 3$	15	$S\emptyset 3$
16	HV_CONT	16	HV_CONT
17	$R\emptyset 2HV\_N$	17	$R\emptyset 2HV\_N$
18	GND	18	GND
19	$R\emptyset 2HV$	19	$R\emptyset 2HV$
20	$R\emptyset 2$	20	$R\emptyset 2$
21	$R\emptyset 2$	21	$R\emptyset 2$
22	$R\emptyset 1$	22	$R\emptyset 1$
23	GND	23	GND
24	$I\emptyset 3$	24	$I\emptyset 3$
25	$I\emptyset 2$	25	$I\emptyset 2$



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 11/81  
Filename: ASTCDetailedDesign

**244.620 - PIGTAIL GLENAIR MWDM-2-L-25-P-6-J-7-8-M  
(24 AWG TEFLON INSULATED WIRES)  
LENGTH 8"**

ASTCE 244.500 J4		ASTC EXTERNAL CONNECTOR	
PIN	SIGNAL	PIN	SIGNAL
1	Din+	9	Din+
2	Sin+	10	Sin+
3	GND	23	GND
4	Sout-	11	Sout-
5	Dout-	12	Dout-
6	Din-	21	Din-
7	Sin-	22	Sin-
8	Sout+	24	Sout+
9	Dout+	25	Dout+
ASTCE 244.500 J5			
1	+ 12 Vdc	6	+ 12 Vdc
2	+ 12 Vdc	-	-
3	+ 12 Vdc	-	-
4	+ 3.3 Vdc	3	+ 3.3 Vdc
5	GND	4	GND
6	GND	17	GND
7	- 8 Vdc	2	- 8 Vdc
8	- 8 Vdc	-	-
9	+ 12 Vdc	19	+ 12 Vdc
10	+ 12 Vdc	-	-
11	+ 12 Vdc	-	-
12	+ 3.3 Vdc	16	+ 3.3 Vdc
13	GND	15	GND
14	GND	-	-
15	- 8 Vdc	14	- 8 Vdc
-	-	1,5,7,8,13, 18,20	nc



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 12/81  
Filename: ASTCDetailedDesign

## 3. BILL OF MATERIALS

ASTCC, SN#244.300					
Nr.	Qu.	Part	Toll.	PCB footprint	Voltage
1	1	4.7uF	±10%	SM/C_2512	50.0
2	2	4.7uF	±10%	SM/C_1812	12.0
3	1	4.7uF	±10%	SM/C_1812	5.0
4	4	4.7uF	±10%	SM/C_1812	12.0
5	4	2.2uF	±10%	SM/C_1812	50.0
6	2	2.2uF	±10%	SM/C_1812	5.0
7	1	.22uF	±10%	SM/C_1210	50.0
8	1	1uF	±10%	SM/C_1206	50.0
9	1	.22uF	±10%	SM/C_0805	50.0
10	5	.1uF	±10%	SM/C_1206	50.0
11	6	.1uF	±10%	SM/C_0805	12.0
12	7	.1uF	±10%	SM/C_0805	5.0
13	2	10pF	±10%	SM/C_0805	50.0
14	1	2.2pF	±10%	SM/C_0805	50.0
15	2	DZ-5V1		SM/D_1812	
16	1	DZ-47V		SM/D_1812	
17	1	LS4148		SM/D_SOD80	
18	1	COAX CONN.		RF/SMA/V	
19	1	9 PINS MICRO-D CONN.		DSUB_MICRO/.75/VS/9	
20	1	5K	±5%	TERM025	
21	2	4K99	±.1%	SM/R_0805	
22	1	249K	±.5%	SM/R_0805	
23	2	47R	±.1%	SM/R_0805	
24	4	100R	±.1%	SM/R_0805	
25	1	1K0	±.1%	SM/R_0805	
26	1	249R	±.1%	SM/R_0805	
27	2	499R	±.1%	SM/R_0805	
28	1	1K	±.1%	SM/R_0805	
29	1	6K8	±.1%	SM/R_0805	
30	1	15K	±.1%	SM/R_0805	
31	1	39K2	±.1%	SM/R_0805	
32	1	4K7	±.1%	SM/R_0805	
33	1	5K6	±.1%	SM/R_0805	
34	3	10R	±.1%	SM/R_0805	
35	1	100R	±.1%	SM/R_0805	
36	1	121R	±.1%	SM/R_0805	
37	1	80K6	±.1%	SM/R_0805	
38	1	10K	±.1%	SM/R_0805	
39	1	100K	±.1%	SM/R_0805	
40	1	EEV CCD87-00			
41	1	ANALOG DEV. AD8067		SOT23_RT5	
42	1	ANALOG DEV. AD8065		SOT23_RT5	





# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 14/81  
Filename: ASTCDetailedDesign

30	1	DZ-27V	SM/D_1812
31	1	DZ-12V	SM/D_1812
32	1	DZ-4V7	SM/D_1812
33	1	DZ-10V	SM/D_1812
34	1	DZ-3.0V	SM/D_1812
35	1	DZ-2.4V	SM/D_1812
36	1	COAX CONN.	RF/SMA/V
37	2	9 PINS MICRO-D CONN.	DSUB_MICRO/.75/VS/9
38	1	25 PINS MICRO-D CONN.	DSUB_MICRO/.75/VS/25
39	1	15 PINS MICRO-D CONN.	DSUB_MICRO/.75/VS/15
40	8	BSS64	SM/SOT23_123
41	2	IRFR5305	TO252AA/DPAK
42	2	BSS82C	SM/SOT23_123
43	2	1K	±.1% SM/R_1206
44	2	47K	±.1% SM/R_1206
45	10	100K	±.1% SM/R_1206
46	3	22R	±.1% SM/R_1206
47	9	10K	±.1% SM/R_1206
48	2	47R	±.1% SM/R_0805
49	3	249R	±.1% SM/R_0805
50	1	100K	±.1% SM/R_0805
51	7	499R	±.1% SM/R_0805
52	2	1K0	±.1% SM/R_0805
53	1	100R	±.1% SM/R_0805
54	3	1K	±.1% SM/R_0805
55	2	10K	±.1% SM/R_0805
56	2	10R	±.1% SM/R_0805
57	10	0R0	±1% SM/R_0805
58	2	249K	±.1% SM/R_0805
59	1	22R	±1% SM/R_0805
60	2	10R	±.1% SM/R_0805
61	1	18K	±.1% SM/R_0805
62	1	2K5	±.1% SM/R_0805
63	1	1K	±.1% SM/R_0805
64	1	66K	±.1% SM/R_0805
65	1	33K	±.1% SM/R_0805
66	1	10K	±.1% SM/R_0805
67	1	68K	±.1% SM/R_0805
68	8	15K	±.1% SM/R_0805
69	1	3K	±.1% SM/R_0805
70	2	1K	±.1% SOG/R_PACK4
71	3	221R	±.1% SOG/R_PACK4
72	2	ANALOG DEV. AD8065	SOT23_RT5
73	1	ANALOG DEV. ADG751	SOT23-TR6
74	1	BURR BROWN ADS803	SOG.025/28/WG.300/L.400
75	1	VT16244	SOG.020/48/WG.300/L.500
76	1	ACTEL A54SX32A	QUAD.50M/208/WG30.60
77	1	TEXAS INSTR. SN65LVDS390	SOG.050/16/WG.244/L.400



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 15/81  
Filename: ASTCDetailedDesign

78	1	TEXAS INSTR. SN65LVDS391	SOG.050/16/WG.244/L.400
79	1	C-MAC CFPS-80M	SM/IQXO-71
80	1	TEXAS INSTR. UCC27424	DIP.100/8/W.300/L.400
81	1	MAXIM MAX1281	SOG.65M/20/TSSOP
82	3	LM1117-5.0	TO252AA/DPAK
83	1	LM1117-3.3	TO252AA/DPAK
84	1	MC79M05CDT	TO252AA/DPAK
85	1	MC79M08CDT	TO252AA/DPAK

## 4. VHDL

```
-- CAMERA AMICA - FPGA - version 04/09/2006

-- top_level.vhd

library ieee;
use ieee.std_logic_1164.all;

entity top_level is
  port (
    qs_p          : out  std_logic_vector( 3 downto 0);
    hk_p          : out  std_logic_vector( 13 downto 0);

    st15_p        : out  std_logic;
    ck40_p        : in   std_logic;
    rst_p         : in   std_logic;
    ind0_p        : in   std_logic;      -- dato da ADC (LSB)
    ind1_p        : in   std_logic;      -- dato da ADC
    ind2_p        : in   std_logic;      -- dato da ADC
    ind3_p        : in   std_logic;      -- dato da ADC
    ind4_p        : in   std_logic;      -- dato da ADC
    ind5_p        : in   std_logic;      -- dato da ADC
    ind6_p        : in   std_logic;      -- dato da ADC
    ind7_p        : in   std_logic;      -- dato da ADC
    ind8_p        : in   std_logic;      -- dato da ADC
    ind9_p        : in   std_logic;      -- dato da ADC
    ind10_p       : in   std_logic;      -- dato da ADC
    ind11_p       : in   std_logic;      -- dato da ADC (MSB)
    din_p         : in   std_logic;      -- VLDS da CPU (set guadagno e
                                         sync CCD)
    sin_p         : in   std_logic;      -- VLDS da CPU
    ckricin_p     : in   std_logic;      -- Ck ritardato per VLDS
    latchuperr_p  : in   std_logic;      -- segnale di Latchup error
    hkdoutin_p   : in   std_logic;      -- input seriale dall'ADC HK
    sstrb_p       : in   std_logic;      -- input strobe dall'ADC HK
    r01_p         : out  std_logic;      -- pilotaggio fasi CCD
    r02_p         : out  std_logic;      -- pilotaggio fasi CCD
    r03_p         : out  std_logic;      -- pilotaggio fasi CCD
    i01_p         : out  std_logic;      -- pilotaggio fasi CCD
    i02_p         : out  std_logic;      -- pilotaggio fasi CCD
    i03_p         : out  std_logic;      -- pilotaggio fasi CCD
    s01_p         : out  std_logic;      -- pilotaggio fasi CCD
    s02_p         : out  std_logic;      -- pilotaggio fasi CCD
    s03_p         : out  std_logic;      -- pilotaggio fasi CCD
    r0_p          : out  std_logic;      -- pilotaggio fasi CCD
    clmp_p        : out  std_logic;      -- pilotaggio clamp CCD
    acq_p         : out  std_logic;      -- strobe lettura ADC
    r02hv_p       : out  std_logic;      -- clock generazione HVDC per
                                         guadagno CCD
    r02hvn_p      : out  std_logic;      -- clock generazione HVDC per
                                         guadagno CCD
    ck1p6_p       : out  std_logic;      -- clock generazione HVDC per
                                         bias CCD
    ck1p6n_p      : out  std_logic;      -- clock generazione HVDC per
```

```

bias CCD
      l_timer_p      : out    std_logic;      -- clock risveglio controllo
latchup
      dout_p        : out    std_logic;      -- VLDS vs CPU (dati e
contr.immagine CCD)
      sout_p        : out    std_logic;      -- VLDS vs CPU
      hvref_p       : out    std_logic;      -- controllo HV per guadagno
CCD
      ckricout_p    : out    std_logic;      -- clock LVDS da CPU da
ritardare
      hkdinout_p   : out    std_logic;      -- set ADC per HK
      sclkout_p    : out    std_logic;      -- clock per ADC di HK
      csout_p       : out    std_logic );     -- cs per ADC di HK

attribute alspin : string;
attribute alspin of ck40_p      : signal is "141";
attribute alspin of rst_p       : signal is "156";
attribute alspin of ind0_p      : signal is "38";
attribute alspin of ind1_p      : signal is "39";
attribute alspin of ind2_p      : signal is "42";
attribute alspin of ind3_p      : signal is "43";
attribute alspin of ind4_p      : signal is "44";
attribute alspin of ind5_p      : signal is "45";
attribute alspin of ind6_p      : signal is "46";
attribute alspin of ind7_p      : signal is "47";
attribute alspin of ind8_p      : signal is "48";
attribute alspin of ind9_p      : signal is "49";
attribute alspin of ind10_p     : signal is "50";
attribute alspin of ind11_p     : signal is "51";
attribute alspin of din_p       : signal is "106";
attribute alspin of sin_p       : signal is "107";
attribute alspin of ckricin_p   : signal is "138";
attribute alspin of latchuperr_p: signal is "158";
attribute alspin of hkdoutin_p  : signal is "57";
attribute alspin of sstrb_p     : signal is "56";
attribute alspin of r01_p       : signal is "91";
attribute alspin of r02_p       : signal is "90";
attribute alspin of r03_p       : signal is "88";
attribute alspin of i01_p       : signal is "97";
attribute alspin of i02_p       : signal is "96";
attribute alspin of i03_p       : signal is "94";
attribute alspin of s01_p       : signal is "81";
attribute alspin of s02_p       : signal is "84";
attribute alspin of s03_p       : signal is "85";
attribute alspin of r0_p        : signal is "93";
attribute alspin of clmp_p     : signal is "19";
attribute alspin of acq_p       : signal is "20";
attribute alspin of r02hv_p     : signal is "87";
attribute alspin of r02hvn_p    : signal is "86";
attribute alspin of ck1p6_p     : signal is "206";
attribute alspin of ck1p6n_p    : signal is "204";
attribute alspin of l_timer_p   : signal is "192";
attribute alspin of dout_p      : signal is "109";
attribute alspin of sout_p      : signal is "108";
attribute alspin of hvref_p     : signal is "207";
attribute alspin of ckricout_p  : signal is "139";
attribute alspin of hkdinout_p  : signal is "55";
attribute alspin of sclkout_p   : signal is "53";
attribute alspin of csout_p     : signal is "54";

```

```

end top_level;

architecture arc_top_level of top_level is

component ccd_driver
  port ( ck40      : in      std_logic;
         rst       : in      std_logic;
         start    : in      std_logic;
         r01      : out     std_logic;
         r02      : out     std_logic;
         r03      : out     std_logic;
         i01      : out     std_logic;
         i02      : out     std_logic;
         i03      : out     std_logic;
         s01      : out     std_logic;
         s02      : out     std_logic;
         s03      : out     std_logic;
         r0       : out     std_logic;
         clmp    : out     std_logic;
         acq     : out     std_logic;
         r02hv   : out     std_logic;
         r02hvn  : out     std_logic;
         fd      : out     std_logic;
         ck2p5   : out     std_logic;
         syncerr : out     std_logic;
         st      : out    std_logic_vector (18 downto 0);
         ck1p6   : out     std_logic;
         ck1p6n  : out     std_logic;
         ck2000  : out     std_logic;
         h       : out    std_logic_vector (2 downto 0);
         f       : out    std_logic_vector(6 downto 0) );
  end component;

component err_complete
  port ( syncerr   : in      std_logic;
         latchuperr : in      std_logic;
         parityerr  : in      std_logic;
         st         : in    std_logic_vector (18 downto 0);
         err        : out    std_logic_vector (13 downto 0) );
  end component;

component rx_complete
  port ( din       : in      std_logic;
         sin       : in      std_logic;
         ck        : in      std_logic;
         rst       : in      std_logic;
         parityerr : out     std_logic;
         csetup    : out    std_logic_vector (13 downto 0);
         start    : out     std_logic;
         hvref    : out     std_logic;
         ckric   : out     std_logic;
         st       : in    std_logic_vector (18 downto 0);
         f        : in    std_logic_vector (6 downto 0) );
  end component;

component hk_complete
  port ( qs        : out    std_logic_vector (3 downto 0);

```

```

      rst      : in      std_logic;
      hkdout   : in      std_logic;
      sstrb    : in      std_logic;
      st       : in      std_logic_vector (18 downto 0);
      ck       : in      std_logic;
      hk       : out     std_logic_vector (13 downto 0);
      hkdin   : out     std_logic;
      sclk    : out     std_logic;
      cs       : out     std_logic;
      h        : in      std_logic_vector (2 downto 0) );
end component;

component tx_complete
  port ( rst      : in      std_logic;
         datain   : in      std_logic_vector (13 downto 0);
         hk       : in      std_logic_vector (13 downto 0);
         err      : in      std_logic_vector (13 downto 0);
         csetup   : in      std_logic_vector (13 downto 0);
         fd       : in      std_logic;
         st       : in      std_logic_vector (18 downto 0);
         ck       : in      std_logic;
         ck40    : in      std_logic;
         dout    : out     std_logic;
         sout    : out     std_logic );
end component;

component inbuf
  port ( pad      : in      std_logic;
         y        : out    std_logic );
end component;

component outbuf
  port ( d       : in      std_logic;
         pad     : out    std_logic );
end component;

signal  ck40      : std_logic;
signal  rst       : std_logic;
signal  datain   : std_logic_vector (13 downto 0);
signal  latchuperr : std_logic;
signal  din       : std_logic;
signal  sin       : std_logic;
signal  ckricin  : std_logic;
signal  hkdoutin : std_logic;
signal  sstrb    : std_logic;
signal  r01      : std_logic;
signal  r02      : std_logic;
signal  r03      : std_logic;
signal  i01      : std_logic;
signal  i02      : std_logic;
signal  i03      : std_logic;
signal  s01      : std_logic;
signal  s02      : std_logic;
signal  s03      : std_logic;
signal  r0       : std_logic;
signal  clmp    : std_logic;
signal  acq     : std_logic;
signal  r02hv   : std_logic;
signal  r02hvn  : std_logic;

```

```

signal      fd          : std_logic;
signal      ck2p5       : std_logic;
signal      syncerr     : std_logic;
signal      st          : std_logic_vector (18 downto 0);
signal      ck1p6       : std_logic;
signal      ck1p6n      : std_logic;
signal      l_timer     : std_logic;
signal      dout         : std_logic;
signal      sout         : std_logic;
signal      hvref        : std_logic;
signal      ckricout    : std_logic;
signal      hkdinout    : std_logic;
signal      sclkout     : std_logic;
signal      csout        : std_logic;
signal      start        : std_logic;
signal      parityerr   : std_logic;
signal      err          : std_logic_vector (13 downto 0);
signal      csetup       : std_logic_vector (13 downto 0);
signal      hk          : std_logic_vector (13 downto 0);
signal      h           : std_logic_vector (2 downto 0);
signal      f            : std_logic_vector (6 downto 0);

signal      qs           : std_logic_vector (3 downto 0);

begin

inst1 : ccd_driver
  port map ( ck40=>ck40, rst=>rst,
             start=>start,
             r01=>r01, r02=>r02, r03=>r03,
             i01=>i01, i02=>i02, i03=>i03,
             s01=>s01, s02=>s02, s03=>s03, r0=>r0,
             clmp=>clmp, acq=>acq, r02hv=>r02hv, r02hvn=>r02hvn,
             fd=>fd, ck2p5=>ck2p5, syncerr=>syncerr,
             st(18 downto 0)=>st(18 downto 0), ck1p6=>ck1p6, ck1p6n=>ck1p6n,
             ck2000=>l_timer, h(2 downto 0)=>h(2 downto 0),
             f(6 downto 0)=>f(6 downto 0) );

inst2 : err_complete
  port map ( syncerr=>syncerr, latchuperr=>latchuperr, parityerr=>parityerr,
             st(18 downto 0)=>st(18 downto 0),
             err(13 downto 0)=>err(13 downto 0) );

inst3 : rx_complete
  port map ( din=>din, sin=>sin, ck=>ckricin, rst=>rst,
             parityerr=>parityerr,
             csetup(13 downto 0)=>csetup(13 downto 0), start=>start,
             hvref=>hvref, ckric=>ckricout,
             st(18 downto 0)=>st(18 downto 0),
             f(6 downto 0)=>f(6 downto 0) );

inst4 : hk_complete
  port map ( qs(3 downto 0)=>qs(3 downto 0),
             rst=>rst,
             hkdout=>hkdoutin, sstrb=>sstrb,
             st(18 downto 0)=>st(18 downto 0),
             ck=>ck1p6, hk(13 downto 0)=>hk(13 downto 0),
             hkdin=>hkdinout, sclk=>sclkout, cs=>csout,
             h(2 downto 0)=>h(2 downto 0));

inst5 : tx_complete
  port map ( rst=>rst,
             datain(13 downto 0)=>datain(13 downto 0),

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 21/81  
Filename: ASTCDetailedDesign

```
    hk(13 downto 0)=>hk(13 downto 0),
    err(13 downto 0)=>err(13 downto 0),
    csetup(13 downto 0)=>csetup(13 downto 0),
    fd=>fd, st(18 downto 0)=>st(18 downto 0),
    ck=>ck2p5, ck40=>ck40, dout=>dout, sout=>sout );

inst6 : inbuf
    port map ( pad=>ck40_p, y=>ck40 );
inst7 : inbuf
    port map ( pad=>rst_p, y=>rst );
inst8 : inbuf
    port map ( pad=>ind0_p, y=>datain(0) );
inst9 : inbuf
    port map ( pad=>ind1_p, y=>datain(1) );
inst10 : inbuf
    port map ( pad=>ind2_p, y=>datain(2) );
inst11 : inbuf
    port map ( pad=>ind3_p, y=>datain(3) );
inst12 : inbuf
    port map ( pad=>ind4_p, y=>datain(4) );
inst13 : inbuf
    port map ( pad=>ind5_p, y=>datain(5) );
inst14 : inbuf
    port map ( pad=>ind6_p, y=>datain(6) );
inst15 : inbuf
    port map ( pad=>ind7_p, y=>datain(7) );
inst16 : inbuf
    port map ( pad=>ind8_p, y=>datain(8) );
inst17 : inbuf
    port map ( pad=>ind9_p, y=>datain(9) );
inst18 : inbuf
    port map ( pad=>ind10_p, y=>datain(10) );
inst19 : inbuf
    port map ( pad=>ind11_p, y=>datain(11) );
inst20 : inbuf
    port map ( pad=>latchuperr_p, y=>latchuperr );
inst21 : inbuf
    port map ( pad=>din_p, y=>din );
inst22 : inbuf
    port map ( pad=>sin_p, y=>sin );
inst23 : inbuf
    port map ( pad=>ckricin_p, y=>ckricin );
inst24 : inbuf
    port map ( pad=>hkdoutin_p, y=>hkdoutin );
inst25 : inbuf
    port map ( pad=>sstrb_p, y=>sstrb );
inst26 : outbuf
    port map ( d=>r01, pad=>r01_p );
inst27 : outbuf
    port map ( d=>r02, pad=>r02_p );
inst28 : outbuf
    port map ( d=>r03, pad=>r03_p );
inst29 : outbuf
    port map ( d=>i01, pad=>i01_p );
inst30 : outbuf
    port map ( d=>i02, pad=>i02_p );
inst31 : outbuf
    port map ( d=>i03, pad=>i03_p );
inst32 : outbuf
    port map ( d=>s01, pad=>s01_p );
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 22/81  
Filename: ASTCDetailedDesign

```
inst33 : outbuf
    port map ( d=>s02, pad=>s02_p );
inst34 : outbuf
    port map ( d=>s03, pad=>s03_p );
inst35 : outbuf
    port map ( d=>r0, pad=>r0_p );
inst36 : outbuf
    port map ( d=>clmp, pad=>clmp_p );
inst37 : outbuf
    port map ( d=>acq, pad=>acq_p );
inst38 : outbuf
    port map ( d=>r02hv, pad=>r02hv_p );
inst39 : outbuf
    port map ( d=>r02hvn, pad=>r02hvn_p );
inst40 : outbuf
    port map ( d=>ck1p6, pad=>ck1p6_p );
inst41 : outbuf
    port map ( d=>ck1p6n, pad=>ck1p6n_p );
inst42 : outbuf
    port map ( d=>l_timer, pad=>l_timer_p );
inst43 : outbuf
    port map ( d=>dout, pad=>dout_p );
inst44 : outbuf
    port map ( d=>sout, pad=>sout_p );
inst45 : outbuf
    port map ( d=>hvref, pad=>hvref_p );
inst46 : outbuf
    port map ( d=>ckricout, pad=>ckricout_p );
inst47 : outbuf
    port map ( d=>hkдин, pad=>hkдин_p );
inst48 : outbuf
    port map ( d=>sclkout, pad=>sclkout_p );
inst49 : outbuf
    port map ( d=>csout, pad=>csout_p );

inst50 : outbuf
    port map ( d=>st(15), pad=>st15_p );

datain(12) <= '0';
datain(13) <= '0';

inst60 : outbuf
    port map ( d=>hk(13), pad=>hk_p(13));
inst61 : outbuf
    port map ( d=>hk(12), pad=>hk_p(12));
inst62 : outbuf
    port map ( d=>hk(11), pad=>hk_p(11));
inst63 : outbuf
    port map ( d=>hk(10), pad=>hk_p(10));
inst64 : outbuf
    port map ( d=>hk(9), pad=>hk_p(9));
inst65 : outbuf
    port map ( d=>hk(8), pad=>hk_p(8));
inst66 : outbuf
    port map ( d=>hk(7), pad=>hk_p(7));
inst67 : outbuf
    port map ( d=>hk(6), pad=>hk_p(6));
inst68 : outbuf
```

```

    port map ( d=>hk(5), pad=>hk_p(5));
inst69 : outbuf
    port map ( d=>hk(4), pad=>hk_p(4));
inst70 : outbuf
    port map ( d=>hk(3), pad=>hk_p(3));
inst71 : outbuf
    port map ( d=>hk(2), pad=>hk_p(2));
inst72 : outbuf
    port map ( d=>hk(1), pad=>hk_p(1));
inst73 : outbuf
    port map ( d=>hk(0), pad=>hk_p(0));

inst80: outbuf
    port map ( d=>qs(0), pad=>qs_p(0));
inst81: outbuf
    port map ( d=>qs(1), pad=>qs_p(1));
inst82: outbuf
    port map ( d=>qs(2), pad=>qs_p(2));
inst83: outbuf
    port map ( d=>qs(3), pad=>qs_p(3));

end arc_top_level;

```

#### 4.1.1. ccd\_driver.vhd

```
-- ccd_driver.vhd

library ieee;
use ieee.std_logic_1164.all;

entity ccd_driver is
  port ( ck40      : in  std_logic; -- clock 40 MHz
         rst       : in  std_logic;
         start     : in  std_logic;
         r01       : out std_logic;
         r02       : out std_logic;
         r03       : out std_logic;
         i01       : out std_logic;
         i02       : out std_logic;
         i03       : out std_logic;
         s01       : out std_logic;
         s02       : out std_logic;
         s03       : out std_logic;
         r0        : out std_logic;
         clmp     : out std_logic;
         acq      : out std_logic;
         r02hv    : out std_logic;
         r02hvn   : out std_logic;
         fd       : out std_logic;
         ck2p5    : out std_logic;
         syncerr  : out std_logic;
         st       : out std_logic_vector (18 downto 0);
         ck1p6    : out std_logic;
         ck1p6n   : out std_logic;
```

```

      ck2000    : out    std_logic;
      h         : out    std_logic_vector (2 downto 0);
      f         : out    std_logic_vector(6 downto 0) );
end ccd_driver;

architecture arc_ccd_driver of ccd_driver is

component gen_buff
  port ( p10       : in     std_logic;
         ck        : in     std_logic; -- clock 20 MHz
         st        : in     std_logic_vector (18 downto 0);
         ck40      : in     std_logic;
         enfd      : in     std_logic;
         r01       : out    std_logic;
         r02       : out    std_logic;
         r03       : out    std_logic;
         i01       : out    std_logic;
         i02       : out    std_logic;
         i03       : out    std_logic;
         s01       : out    std_logic;
         s02       : out    std_logic;
         s03       : out    std_logic;
         r0        : out    std_logic;
         clmp      : out    std_logic;
         acq       : out    std_logic;
         r02hv     : out    std_logic;
         r02hvn    : out    std_logic;
         fd        : out    std_logic;
         ck2p5     : out    std_logic); -- pixel clock
end component;

component div
  port ( ck        : in     std_logic;   -- clock 40 MHz
         rst       : in     std_logic;
         ck_4      : out    std_logic;   -- clock 20 MHz
         ck_24     : out    std_logic;   -- clock 2 MHz
         f         : out    std_logic_vector(6 downto 0) );
end component;

component count_ck
  port ( cl        : in     std_logic;
         ck        : in     std_logic;
         rst       : in     std_logic;
         q         : out   std_logic_vector (14 downto 0) );
end component;

component count_row
  port ( cl        : in     std_logic;
         ck        : in     std_logic;
         rst       : in     std_logic;
         r         : out   std_logic_vector (9 downto 0);
         enfd     : out   std_logic );
end component;

component ccd_fsm
  port ( q         : in     std_logic_vector (14 downto 0);
         r         : in     std_logic_vector (9 downto 0);
         ck       : in     std_logic;
         start    : in     std_logic;

```

```

      rst      : in      std_logic;
      state    : out     std_logic_vector (18 downto 0);
      p10     : out     std_logic;
      syncerr  : out     std_logic;
      ck2000   : out     std_logic;
      h        : out     std_logic_vector (2 downto 0) );
end component;

signal      ck_4      : std_logic;
signal      ck_24     : std_logic;
signal      p10       : std_logic;
signal      q         : std_logic_vector (14 downto 0);
signal      r         : std_logic_vector (9 downto 0);
signal      enfd      : std_logic;
signal      s         : std_logic_vector (18 downto 0);
signal      fd1       : std_logic;

begin

Inst1 : gen_buff
  port map ( p10=>p10, ck=>ck_4, st(18 downto 0)=>s(18 downto 0),
             ck40=>ck40, enfd=>enfd, r01=>r01, r02=>r02,
             r03=>r03, i01=>i01, i02=>i02, i03=>i03,
             s01=>s01, s02=>s02, s03=>s03, r0=>r0,
             clmp=>clmp, acq=>acq, r02hv=>r02hv, r02hvn=>r02hvn,
             fd=>fd1, ck2p5=>ck2p5 );
Inst2 : div
  port map ( ck=>ck40, rst=>rst, ck_4=>ck_4, ck_24=>ck_24,
              f(6 downto 0)=>f(6 downto 0) );
Inst3 : count_ck
  port map ( cl=>p10, ck=>ck_4, rst=>rst,
              q(14 downto 0)=>q(14 downto 0) );
Inst4 : count_row
  port map ( cl=>s(3), ck=>s(12), rst=>rst,
              r(9 downto 0)=>r(9 downto 0), enfd=>enfd );
Inst5 : ccd_fsm
  port map ( q(14 downto 0)=>q(14 downto 0),
              r(9 downto 0)=>r(9 downto 0),
              ck=>ck_4, start=>start, rst=>rst,
              state(18 downto 0)=>s(18 downto 0), p10=> p10,
              syncerr=>syncerr, ck2000 =>ck2000,
              h(2 downto 0)=>h(2 downto 0) );

st <= s;
ck1p6 <= ck_24;
ck1p6n <= not ck_24;
fd <= fd1;

end arc_ccd_driver;

```

#### 4.1.2. gen\_buff.vhd

```
-- gen_buff.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

```

use ieee.std_logic_unsigned.all;

entity gen_buff is
  port ( p10      : in  std_logic;
         ck       : in  std_logic;    -- 10 MHz
         st       : in  std_logic_vector (18 downto 0);
         ck40    : in  std_logic;    -- 40 MHz
         enfd    : in  std_logic;
         r01     : out std_logic;
         r02     : out std_logic;
         r03     : out std_logic;
         i01     : out std_logic;
         i02     : out std_logic;
         i03     : out std_logic;
         s01     : out std_logic;
         s02     : out std_logic;
         s03     : out std_logic;
         r0      : out std_logic;
         clmp   : out std_logic;
         acq    : out std_logic;
         r02hv  : out std_logic;
         r02hvn : out std_logic;
         fd     : out std_logic;
         ck2p5  : out std_logic );
end gen_buff;

architecture arc_gen_buff of gen_buff is

  constant maxcount      : unsigned(5 downto 0):="100111";

  signal      r1      : std_logic;
  signal      r2      : std_logic;
  signal      r3      : std_logic;
  signal      i1      : std_logic;
  signal      i2      : std_logic;
  signal      i3a     : std_logic;
  signal      i3b     : std_logic;
  signal      r1q     : std_logic;
  signal      r2q     : std_logic;
  signal      r3q     : std_logic;
  signal      i1q     : std_logic;
  signal      i2q     : std_logic;
  signal      i3aq    : std_logic;
  signal      i3bq    : std_logic;
  signal      r1s     : std_logic;
  signal      r2s     : std_logic;
  signal      r3s     : std_logic;
  signal      i1s     : std_logic;
  signal      i2s     : std_logic;
  signal      i3s     : std_logic;
  signal      s1s     : std_logic;
  signal      s2s     : std_logic;
  signal      s3s     : std_logic;
  signal      r2hvs   : std_logic;
  signal      fds     : std_logic;
  signal      dec     : std_logic;

  signal      a0      : std_logic;

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 27/81  
Filename: ASTCDetailedDesign

```
signal      a1      : std_logic;
signal      a2      : std_logic;
signal      a3      : std_logic;
signal      a4      : std_logic;

signal      s1      : std_logic;
signal      s2      : std_logic;
signal      s3      : std_logic;
signal      s4      : std_logic;
signal      s5      : std_logic;
signal      s6      : std_logic;
signal      s7      : std_logic;
signal      s8      : std_logic;

signal      rrq     : std_logic_vector (2 downto 0);
signal      rq      : std_logic_vector (2 downto 0);
signal      q       : std_logic_vector (5 downto 0);
signal      qu      : unsigned(5 downto 0);

signal      st5q    : std_logic;
signal      st0q    : std_logic;
signal      st1q    : std_logic;
signal      st2q    : std_logic;

begin

process(ck, p10, dec)
begin
  if (ck'event and ck = '1') then
    if (p10 = '1' or dec = '1') then
      q <= (others => '0');
      else q <= q + '1';
    end if;
  end if;
end process;

process (qu)
begin
  if qu = maxcount then dec <= '1';
  else dec <= '0';
  end if;
end process;

process(ck)
begin
  if (ck'event and ck = '1') then
    r1q <= r1;
    r2q <= r2;
    r3q <= r3;
    i1q <= i1;
    i2q <= i2;
    i3aq <= i3a;
    i3bq <= i3b;

    rq(0) <= r1s;
    rq(1) <= r2s;
    rq(2) <= r3s;
    i01 <= i1s;
    i02 <= i2s;
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 28/81  
Filename: ASTCDetailedDesign

```
i03 <= i3s;
s01 <= s1s;
s02 <= s2s;
s03 <= s3s;

a0 <= not r3q;
a1 <= r2q;

st0q <= st(0);
st1q <= st(1);
st2q <= st(2);
st5q <= st(5);

end if;
end process;

process(ck)
begin
  if (ck'event and ck = '0') then
    a2 <= r2q;
  end if;
end process;

process(ck40)
begin
  if (ck40'event and ck40 = '0') then
    rrq(0) <= rq(0);
    rrq(1) <= rq(1);
    rrq(2) <= rq(2);
    a3 <= a1;
    a4 <= a3;
  end if;
end process;

process(r3q)
begin
  if (r3q'event and r3q = '1') then
    fd <= fds;
  end if;
end process;

qu <= unsigned(q);
r1 <= q(1);
r2 <= not q(0) and not q(1);
r3 <= q(0) and not q(1);
i1 <= not q(4) and not q(5);
i2 <= ((q(2) and q(3)) xor q(4)) and not q(5);
i3a <= (q(3) and q(4) and not q(5)) or (not q(3) and not q(4) and q(5));
i3b <= (not q(2) and not q(3) and not q(4) and not q(5)) or i3a;

r1s <= r1q or s1;
r2s <= (r2q and s2) or s3;
r3s <= r3q and s4;
i1s <= (i1q and s5) or st0q;
i2s <= i2q and s5;
i3s <= (i3aq and st1q) or (i3bq and st(2));
s1s <= (i1q and s6) or st0q;
s2s <= i2q and s6;
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 29/81  
Filename: ASTCDetailedDesign

```
s3s <= (i3aq and s7) or (i3bq and st(2));  
acq <= r3q;  
r2hvs <= a1 or a2;  
fds <= (st(9) or st(10)) and enfd;  
  
s1 <= st(4) or st(5) or st(6) or st(7) or st(12) or st(13);  
s2 <= not(st(7) or st(12) or st(13));  
s3 <= st(4) or st(5) or st(6);  
s4 <= not s1;  
s5 <= st1q or st(2);  
s6 <= st(1) or st(2) or st(5);  
-- s7 <= st(1) or st(5);  
s7 <= st1q or st5q;  
s8 <= st(0) or st(1) or st(2) or st(3) or st(14) or st(15) or st(16) or  
st(17) or st(18);  
  
r01 <= rrq(0) or rq(0);  
r02 <= rrq(1) or rq(1);  
r03 <= rrq(2) or rq(2);  
  
r0 <= (a4 and not(a1)) or st(3);  
  
clmp <= a0 and not s8;  
  
ck2p5 <= r3q;  
  
r02hv <= a1 or a2;  
r02hvn <= not (a1 or a2);  
  
end arc_gen_buff;
```

### 4.1.3. div.vhd

```
-- div.vhd  
  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
use ieee.std_logic_unsigned.all;  
  
entity div is  
    port ( ck           : in      std_logic;  -- clock 40 MHz  
          rst           : in      std_logic;  
          ck_4          : out     std_logic;  -- clock 10 MHz  
          ck_24         : out     std_logic; -- clock 1.66 MHz  
          f             : out     std_logic_vector(6 downto 0) );  
end div;  
  
architecture arc_div of  div is  
  
constant maxcount  :  unsigned(2 downto 0):="101";  
constant a1        :  unsigned(2 downto 0):="011";  
constant a2        :  unsigned(2 downto 0):="100";  
constant a3        :  unsigned(2 downto 0):="101";  
  
signal q1         :  std_logic_vector(1 downto 0);
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 30/81  
Filename: ASTCDetailedDesign

```
signal q2          : unsigned(2 downto 0);
signal q3          : std_logic_vector(6 downto 0);

begin

process(ck, rst)
begin
  if rst = '0' then
    q1 <= (others => '0');
  elsif (ck'event and ck = '1') then
    q1 <= q1 + '1';
  end if;
end process;

process(q1(1), rst)
begin
  if rst = '0' then
    q2 <= (others => '0');
  elsif (q1(1)'event and q1(1) = '1') then
    if q2 = maxcount then
      q2 <= (others => '0');
    else q2 <= q2 + 1;
    end if;
  end if;
end process;

process(q1(1))
begin
  if (q1(1)'event and q1(1) = '1') then
    if (q2=a1 or q2=a2 or q2=a3) then
      ck_24 <= '1';
    else
      ck_24 <= '0';
    end if;
  end if;
end process;

process(q1(1), rst)
begin
  if rst = '0' then
    q3 <= (others => '0');
  elsif (q1(1)'event and q1(1) = '1') then
    q3 <= q3 + '1';
  end if;
end process;

ck_4 <= q1(1);
f <= q3;

end arc_div;
```

### 4.1.4. count\_ck.vhd

```
-- count_ck.vhd

library ieee;
use ieee.std_logic_1164.all;
```

```

entity count_ck is
  port ( cl      : in    std_logic;
         ck      : in    std_logic;
         rst     : in    std_logic;
         q       : out   std_logic_vector(14 downto 0) );
end count_ck;

architecture arc_count_ck of  count_ck is

  component my_count15
    port ( data      : in    std_logic_vector(14 downto 0);
           enable    : in    std_logic;
           sload    : in    std_logic;
           aclr     : in    std_logic;
           clock    : in    std_logic;
           q        : out   std_logic_vector(14 downto 0) );
  end component;

  signal     a      : std_logic;

begin

  inst1 : my_count15
    port map ( data(14 downto 0)=>"0000000000000000",
               enable=>'1', sload=>cl, aclr=>a, clock=>ck,
               q(14 downto 0)=>q(14 downto 0) );

  a <= not rst;

end arc_count_ck;

```

#### 4.1.5. my\_count15.vhd

```

-- Version: 6.1 6.1.1.24

library ieee;
use ieee.std_logic_1164.all;
library a54SXA;

entity my_count15 is
  port( Data : in std_logic_vector(14 downto 0);Enable, Sload,
        Aclr, Clock : in std_logic; Q : out std_logic_vector(14 downto
        0)) ;
end my_count15;

architecture DEF_ARCH of  my_count15 is

  component BUFF
    port(A : in std_logic;  Y : out std_logic) ;
  end component;

  component DFC1B
    port(D, CLK, CLR : in std_logic;  Q : out std_logic) ;
  end component;

  component AND4

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 32/81  
Filename: ASTCDetailedDesign

```
port(A, B, C, D : in std_logic;  Y : out std_logic) ;
end component;

component XOR2
    port(A, B : in std_logic;  Y : out std_logic) ;
end component;

component AX1C
    port(A, B, C : in std_logic;  Y : out std_logic) ;
end component;

component CM8
    port(D0, D1, D2, D3, S00, S01, S10, S11 : in std_logic;
          Y : out std_logic) ;
end component;

component AND3
    port(A, B, C : in std_logic;  Y : out std_logic) ;
end component;

component INV
    port(A : in std_logic;  Y : out std_logic) ;
end component;

component AND4A
    port(A, B, C, D : in std_logic;  Y : out std_logic) ;
end component;

component AND2
    port(A, B : in std_logic;  Y : out std_logic) ;
end component;

component VCC
    port( Y : out std_logic);
end component;

component GND
    port( Y : out std_logic);
end component;

signal Q_0_net, Q_1_net, Q_2_net, Q_3_net, Q_4_net, Q_5_net,
       Q_6_net, Q_7_net, Q_8_net, Q_9_net, Q_10_net, Q_11_net,
       Q_12_net, Q_13_net, Q_14_net, LdAux_0_net, LdAux_6_net,
       LdAux_12_net, ClrAux_0_net, ClrAux_6_net, ClrAux_12_net,
       En, CI_0_net, RCO_0_net, CI_1_net, RCO_1_net, CI_2_net,
       RCO_2_net, CI_3_net, INV_1_Y, XOR_1_Y, AX1C_4_Y, CM8_6_Y,
       CM8_8_Y, CM8_12_Y, INV_2_Y, XOR_0_Y, AX1C_3_Y, AX1C_0_Y,
       AND4_1_Y, AND4_2_Y, CM8_16_Y, CM8_9_Y, CM8_13_Y, CM8_5_Y,
       CM8_14_Y, INV_3_Y, XOR_2_Y, AX1C_2_Y, AX1C_6_Y, AND4A_0_Y,
       AND4_0_Y, CM8_1_Y, CM8_2_Y, CM8_15_Y, CM8_7_Y, CM8_4_Y,
       INV_0_Y, XOR_3_Y, AX1C_1_Y, AX1C_5_Y, AND4_3_Y, AND4_4_Y,
       CM8_11_Y, CM8_3_Y, CM8_17_Y, CM8_10_Y, CM8_0_Y, VCC_1_net,
       GND_1_net : std_logic ;
begin
    Q(0) <= Q_0_net;
    Q(1) <= Q_1_net;
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 33/81  
Filename: ASTCDetailedDesign

```
Q(2) <= Q_2_net;  
  
Q(3) <= Q_3_net;  
  
Q(4) <= Q_4_net;  
  
Q(5) <= Q_5_net;  
  
Q(6) <= Q_6_net;  
  
Q(7) <= Q_7_net;  
  
Q(8) <= Q_8_net;  
  
Q(9) <= Q_9_net;  
  
Q(10) <= Q_10_net;  
  
Q(11) <= Q_11_net;  
  
Q(12) <= Q_12_net;  
  
Q(13) <= Q_13_net;  
  
Q(14) <= Q_14_net;  
  
VCC_2_net : VCC port map(Y => VCC_1_net);  
GND_2_net : GND port map(Y => GND_1_net);  
BUF_LdAux_6_inst : BUFF  
    port map(A => Sload, Y => LdAux_6_net);  
DFC1B_Q_0_inst : DFC1B  
    port map(D => CM8_2_Y, CLK => Clock, CLR => ClrAux_0_net,  
             Q => Q_0_net);  
AND4_2 : AND4  
    port map(A => Data(8), B => Data(9), C => Data(10), D =>  
             Data(11), Y => AND4_2_Y);  
XOR_1 : XOR2  
    port map(A => Q_12_net, B => Q_13_net, Y => XOR_1_Y);  
AND4_3 : AND4  
    port map(A => Q_4_net, B => Q_5_net, C => Q_6_net, D =>  
             Q_7_net, Y => AND4_3_Y);  
DFC1B_Q_14_inst : DFC1B  
    port map(D => CM8_6_Y, CLK => Clock, CLR => ClrAux_12_net,  
             Q => Q_14_net);  
AX1C_1 : AX1C  
    port map(A => Q_4_net, B => Q_5_net, C => Q_6_net, Y =>  
             AX1C_1_Y);  
CM8_14 : CM8  
    port map(D0 => RCO_2_net, D1 => AND4_1_Y, D2 => AND4_2_Y,  
             D3 => AND4_2_Y, S00 => En, S01 => VCC_1_net, S10 =>  
             LdAux_12_net, S11 => GND_1_net, Y => CM8_14_Y);  
DFC1B_RCO_0_inst : DFC1B  
    port map(D => CM8_4_Y, CLK => Clock, CLR => ClrAux_0_net,  
             Q => RCO_0_net);  
AX1C_3 : AX1C  
    port map(A => Q_8_net, B => Q_9_net, C => Q_10_net, Y =>  
             AX1C_3_Y);  
OR3_2_inst : AND3  
    port map(A => Enable, B => RCO_0_net, C => RCO_1_net, Y =>
```

```

    CI_2_net);
INV_ClrAux_12_inst : INV
  port map(A => Aclr, Y => ClrAux_12_net);
CM8_16 : CM8
  port map(D0 => Q_10_net, D1 => AX1C_3_Y, D2 => Data(10),
            D3 => Data(10), S00 => CI_2_net, S01 => VCC_1_net, S10 =>
            LdAux_12_net, S11 => GND_1_net, Y => CM8_16_Y);
BUF_LdAux_0_inst : BUFF
  port map(A => Sload, Y => LdAux_0_net);
INV_3 : INV
  port map(A => Q_0_net, Y => INV_3_Y);
AND4_0 : AND4
  port map(A => Data(0), B => Data(1), C => Data(2), D =>
            Data(3), Y => AND4_0_Y);
INV_ClrAux_0_inst : INV
  port map(A => Aclr, Y => ClrAux_0_net);
AND4A_0 : AND4A
  port map(A => Q_0_net, B => Q_1_net, C => Q_2_net, D =>
            Q_3_net, Y => AND4A_0_Y);
DFC1B_Q_10_inst : DFC1B
  port map(D => CM8_16_Y, CLK => Clock, CLR => ClrAux_12_net,
            Q => Q_10_net);
DFC1B_Q_2_inst : DFC1B
  port map(D => CM8_1_Y, CLK => Clock, CLR => ClrAux_0_net,
            Q => Q_2_net);
XOR_2 : XOR2
  port map(A => Q_0_net, B => Q_1_net, Y => XOR_2_Y);
INV_1 : INV
  port map(A => Q_12_net, Y => INV_1_Y);
CM8_12 : CM8
  port map(D0 => Q_12_net, D1 => INV_1_Y, D2 => Data(12),
            D3 => Data(12), S00 => CI_3_net, S01 => VCC_1_net, S10 =>
            LdAux_12_net, S11 => GND_1_net, Y => CM8_12_Y);
BUF_LdAux_12_inst : BUFF
  port map(A => Sload, Y => LdAux_12_net);
AND4_1 : AND4
  port map(A => Q_8_net, B => Q_9_net, C => Q_10_net, D =>
            Q_11_net, Y => AND4_1_Y);
AND4_4 : AND4
  port map(A => Data(4), B => Data(5), C => Data(6), D =>
            Data(7), Y => AND4_4_Y);
INV_ClrAux_6_inst : INV
  port map(A => Aclr, Y => ClrAux_6_net);
AX1C_5 : AX1C
  port map(A => Q_5_net, B => Q_6_net, C => Q_7_net, Y =>
            AX1C_5_Y);
CM8_17 : CM8
  port map(D0 => Q_7_net, D1 => AX1C_5_Y, D2 => Data(7),
            D3 => Data(7), S00 => CI_1_net, S01 => Q_4_net, S10 =>
            LdAux_6_net, S11 => GND_1_net, Y => CM8_17_Y);
DFC1B_Q_13_inst : DFC1B
  port map(D => CM8_8_Y, CLK => Clock, CLR => ClrAux_12_net,
            Q => Q_13_net);
CM8_13 : CM8
  port map(D0 => Q_11_net, D1 => AX1C_0_Y, D2 => Data(11),
            D3 => Data(11), S00 => CI_2_net, S01 => Q_8_net, S10 =>
            LdAux_12_net, S11 => GND_1_net, Y => CM8_13_Y);
CM8_1 : CM8
  port map(D0 => Q_2_net, D1 => AX1C_2_Y, D2 => Data(2),
            D3 => Data(2));

```

```

D3 => Data(2), S00 => CI_0_net, S01 => VCC_1_net, S10 =>
  LdAux_0_net, S11 => GND_1_net, Y => CM8_1_Y;
DFC1B_Q_11_inst : DFC1B
  port map(D => CM8_13_Y, CLK => Clock, CLR => ClrAux_12_net,
  Q => Q_11_net);
DFC1B_Q_12_inst : DFC1B
  port map(D => CM8_12_Y, CLK => Clock, CLR => ClrAux_12_net,
  Q => Q_12_net);
DFC1B_Q_6_inst : DFC1B
  port map(D => CM8_11_Y, CLK => Clock, CLR => ClrAux_6_net,
  Q => Q_6_net);
CM8_6 : CM8
  port map(D0 => Q_14_net, D1 => AX1C_4_Y, D2 => Data(14),
  D3 => Data(14), S00 => CI_3_net, S01 => VCC_1_net, S10 =>
  LdAux_12_net, S11 => GND_1_net, Y => CM8_6_Y);
INV_2 : INV
  port map(A => Q_8_net, Y => INV_2_Y);
CM8_7 : CM8
  port map(D0 => Q_1_net, D1 => XOR_2_Y, D2 => Data(1), D3 =>
  Data(1), S00 => CI_0_net, S01 => VCC_1_net, S10 =>
  LdAux_0_net, S11 => GND_1_net, Y => CM8_7_Y);
CM8_2 : CM8
  port map(D0 => Q_0_net, D1 => INV_3_Y, D2 => Data(0), D3 =>
  Data(0), S00 => CI_0_net, S01 => VCC_1_net, S10 =>
  LdAux_0_net, S11 => GND_1_net, Y => CM8_2_Y);
DFC1B_Q_7_inst : DFC1B
  port map(D => CM8_17_Y, CLK => Clock, CLR => ClrAux_6_net,
  Q => Q_7_net);
INVBUFO_0_inst : BUFF
  port map(A => Enable, Y => En);
CM8_10 : CM8
  port map(D0 => Q_5_net, D1 => XOR_3_Y, D2 => Data(5), D3 =>
  Data(5), S00 => CI_1_net, S01 => VCC_1_net, S10 =>
  LdAux_6_net, S11 => GND_1_net, Y => CM8_10_Y);
DFC1B_Q_8_inst : DFC1B
  port map(D => CM8_9_Y, CLK => Clock, CLR => ClrAux_6_net,
  Q => Q_8_net);
AX1C_4 : AX1C
  port map(A => Q_12_net, B => Q_13_net, C => Q_14_net, Y =>
  AX1C_4_Y);
DFC1B_Q_5_inst : DFC1B
  port map(D => CM8_10_Y, CLK => Clock, CLR => ClrAux_6_net,
  Q => Q_5_net);
AX1C_2 : AX1C
  port map(A => Q_0_net, B => Q_1_net, C => Q_2_net, Y =>
  AX1C_2_Y);
DFC1B_Q_1_inst : DFC1B
  port map(D => CM8_7_Y, CLK => Clock, CLR => ClrAux_0_net,
  Q => Q_1_net);
DFC1B_Q_4_inst : DFC1B
  port map(D => CM8_3_Y, CLK => Clock, CLR => ClrAux_0_net,
  Q => Q_4_net);
XOR_0 : XOR2
  port map(A => Q_8_net, B => Q_9_net, Y => XOR_0_Y);
CM8_3 : CM8
  port map(D0 => Q_4_net, D1 => INV_0_Y, D2 => Data(4), D3 =>
  Data(4), S00 => CI_1_net, S01 => VCC_1_net, S10 =>
  LdAux_0_net, S11 => GND_1_net, Y => CM8_3_Y);
CM8_8 : CM8

```



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 36/81  
Filename: ASTCDetailedDesign

```
port map(D0 => Q_13_net, D1 => XOR_1_Y, D2 => Data(13),  
D3 => Data(13), S00 => CI_3_net, S01 => VCC_1_net, S10 =>  
LdAux_12_net, S11 => GND_1_net, Y => CM8_8_Y);  
CM8_5 : CM8  
port map(D0 => Q_9_net, D1 => XOR_0_Y, D2 => Data(9), D3 =>  
Data(9), S00 => CI_2_net, S01 => VCC_1_net, S10 =>  
LdAux_6_net, S11 => GND_1_net, Y => CM8_5_Y);  
CM8_11 : CM8  
port map(D0 => Q_6_net, D1 => AX1C_1_Y, D2 => Data(6),  
D3 => Data(6), S00 => CI_1_net, S01 => VCC_1_net, S10 =>  
LdAux_6_net, S11 => GND_1_net, Y => CM8_11_Y);  
CM8_15 : CM8  
port map(D0 => Q_3_net, D1 => AX1C_6_Y, D2 => Data(3),  
D3 => Data(3), S00 => CI_0_net, S01 => Q_0_net, S10 =>  
LdAux_0_net, S11 => GND_1_net, Y => CM8_15_Y);  
AX1C_0 : AX1C  
port map(A => Q_9_net, B => Q_10_net, C => Q_11_net, Y =>  
AX1C_0_Y);  
OR4_3_inst : AND4  
port map(A => Enable, B => RCO_0_net, C => RCO_1_net, D =>  
RCO_2_net, Y => CI_3_net);  
INVBUF1_0_inst : BUFF  
port map(A => Enable, Y => CI_0_net);  
DFC1B_Q_9_inst : DFC1B  
port map(D => CM8_5_Y, CLK => Clock, CLR => ClrAux_6_net,  
Q => Q_9_net);  
INV_0 : INV  
port map(A => Q_4_net, Y => INV_0_Y);  
CM8_0 : CM8  
port map(D0 => RCO_1_net, D1 => AND4_3_Y, D2 => AND4_4_Y,  
D3 => AND4_4_Y, S00 => En, S01 => VCC_1_net, S10 =>  
LdAux_6_net, S11 => GND_1_net, Y => CM8_0_Y);  
CM8_4 : CM8  
port map(D0 => RCO_0_net, D1 => AND4A_0_Y, D2 => AND4_0_Y,  
D3 => AND4_0_Y, S00 => En, S01 => VCC_1_net, S10 =>  
LdAux_0_net, S11 => GND_1_net, Y => CM8_4_Y);  
CM8_9 : CM8  
port map(D0 => Q_8_net, D1 => INV_2_Y, D2 => Data(8), D3 =>  
Data(8), S00 => CI_2_net, S01 => VCC_1_net, S10 =>  
LdAux_6_net, S11 => GND_1_net, Y => CM8_9_Y);  
XOR_3 : XOR2  
port map(A => Q_4_net, B => Q_5_net, Y => XOR_3_Y);  
DFC1B_Q_3_inst : DFC1B  
port map(D => CM8_15_Y, CLK => Clock, CLR => ClrAux_0_net,  
Q => Q_3_net);  
AX1C_6 : AX1C  
port map(A => Q_1_net, B => Q_2_net, C => Q_3_net, Y =>  
AX1C_6_Y);  
OR2_1_inst : AND2  
port map(A => Enable, B => RCO_0_net, Y => CI_1_net);  
DFC1B_RCO_1_inst : DFC1B  
port map(D => CM8_0_Y, CLK => Clock, CLR => ClrAux_6_net,  
Q => RCO_1_net);  
DFC1B_RCO_2_inst : DFC1B  
port map(D => CM8_14_Y, CLK => Clock, CLR => ClrAux_12_net,  
Q => RCO_2_net);  
end DEF ARCH;
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 37/81  
Filename: ASTCDetailedDesign

### 4.1.6. count\_row.vhd

```
-- count_row.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity count_row is
    port ( cl      : in    std_logic;
           ck      : in    std_logic;
           rst     : in    std_logic;
           r       : out   std_logic_vector (9 downto 0);
           enfd   : out   std_logic );
end count_row;

architecture arc_count_row of count_row is

    signal q1      : std_logic;
    signal clr     : std_logic;
    signal q       : unsigned(9 downto 0);

    constant a : unsigned(9 downto 0):= "0000000001" ;

begin

begin

process(ck, clr)
begin
    if clr = '1' then
        q <= (others => '0');
    elsif (ck'event and ck = '1') then
        q <= q + 1;
    end if;
end process;

process(q1, clr)
begin
    if clr = '1' then
        enfd <= '0';
    elsif (q1'event and q1 = '1') then
        enfd <= '1';
    end if;
end process;

q1 <= '1' when (q = a) else '0';

r <= std_logic_vector(q);
clr <= cl or not rst;

end arc_count_row;
```

### 4.1.7. ccd\_fsm.vhd

```
-- ccd_fsm.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ccd_fsm is
  port ( q      : in  std_logic_vector (14 downto 0);
         r      : in  std_logic_vector (9 downto 0);
         ck     : in  std_logic;      -- 10 MHz
         start  : in  std_logic;
         rst    : in  std_logic;
         state  : out std_logic_vector (18 downto 0);
         p10   : out std_logic;
         syncerr: out std_logic;
         ck2000: out std_logic;
         h     : out std_logic_vector (2 downto 0) );
end ccd_fsm;

architecture arc_ccd_fsm of ccd_fsm is

component decod
  generic (wid : integer);
  port ( a      : in  std_logic_vector (wid downto 0);
         n      : in  integer;
         dec    : out std_logic );
end component;

constant a1      : integer := 21399;  -- ***** 21399
constant a2      : integer := 39;
constant a3      : integer := 2207;   -- ***** 2207
constant a4      : integer := 159;
constant a5      : integer := 1999;
constant a6      : integer := 519;
constant a7      : integer := 512;    -- ***** 512 righe *****
constant wid9   : integer := 9;
constant wid14  : integer := 14;

signal sa1      : std_logic;  -- dec21399
signal sa2      : std_logic;  -- dec39
signal sa3      : std_logic;  -- dec2207
signal sa4      : std_logic;  -- dec159
signal sa5      : std_logic;  -- dec1999
signal sa6      : std_logic;  -- dec519
signal sa7      : std_logic;  -- dec511
signal sa2q     : std_logic;
signal ckfsm    : std_logic;
signal cl fsm   : std_logic;
signal clstart  : std_logic;
signal strans   : std_logic;
signal stransq  : std_logic;
signal saux     : std_logic;
signal startq   : std_logic;
signal ck18     : std_logic;
-- signal ck1      : std_logic;
signal ck0      : std_logic;
```

```

signal      sync      : std_logic;
signal      ck3       : std_logic;
signal      h3        : unsigned (2 downto 0);

type state_values is (st0, st1, st2, st3, st4, st5, st6, st7,
                      st8, st9, st10, st11, st12, st13, st14,
                      st15, st16, st17, st18);
signal      pres_state, next_state: state_values;
attribute syn_encoding : string;
attribute syn_encoding of pres_state : signal is "safe,onehot";

begin

inst1: decod generic map (wid14) port map(q, a1, sa1);
inst2: decod generic map (wid14) port map(q, a2, sa2);
inst3: decod generic map (wid14) port map(q, a3, sa3);
inst4: decod generic map (wid14) port map(q, a4, sa4);
inst5: decod generic map (wid14) port map(q, a5, sa5);
inst6: decod generic map (wid14) port map(q, a6, sa6);
inst7: decod generic map (wid9)  port map(r, a7, sa7);

-- p10

process (ck, rst)
begin
  if rst = '0' then
    stransq <= '0';
    sa2q <= '0';
  elsif (ck'event and ck = '0') then
    stransq <= strans;
    sa2q <= sa2;
  end if;
end process;

-- fsm clock

process (ck,rst)
begin
  if rst = '0' then
    ckfsm <= '0';
  elsif (ck'event and ck = '1') then
    ckfsm <= saux;
  end if;
end process;

-- start

process (start, clstart)
begin
  if clstart = '1' then
    startq <= '0';
  elsif (start'event and start = '1') then
    startq <= '1';
  end if;
end process;

process (ck, rst)

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 40/81  
Filename: ASTCDetailedDesign

```
begin
  if rst ='0' then
    clstart <= '1';
  elsif (ck'event and ck = '1') then
    clstart <= clfsm;
  end if;
end process;

-- syncerr

process (pres_state)
begin
  if pres_state = st18 then
    ck18 <= '1';
  else
    ck18 <= '0';
  end if;
end process;

process (pres_state)
begin
  if pres_state = st0 then
    ck0 <= '1';
  else
    ck0 <= '0';
  end if;
end process;

process (ck18,pres_state)
begin
  if pres_state = st1 then
    sync <= '1';
  elsif (ck18'event and ck18 = '1') then
    sync <= '0';
  end if;
end process;

process (ck0)
begin
  if (ck0'event and ck0 = '1') then
    syncerr <= sync;
  end if;
end process;

-- ck2000

process (pres_state)
begin
  if pres_state = st4 then
    ck2000 <= '1';
  else
    ck2000 <= '0';
  end if;
end process;

-- fsm register
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 41/81  
Filename: *ASTCDetailedDesign*

```
process (ckfsm, rst)
begin
  if rst = '0' then
    pres_state <= st0;
  elsif (ckfsm'event and ckfsm = '1') then
    if ckfsm = '1' then
      pres_state <= st0;
    else pres_state <= next_state;
      pres_state <= next_state;
    end if;
  end if;
end process;

-- fsm combinational block

process (pres_state, sa1, sa2, sa3, sa4, sa5, sa6, sa7)
begin
  case pres_state is
    when st0 =>
      case sa6 is
        when '1' =>
          next_state <= st1;
          strans <= '1';
        when '0' =>
          next_state <= st0;
          strans <= '0';
        when others => next_state <= st0;
      end case;
    when st1 =>
      case sa2 is
        when '1' =>
          next_state <= st2;
          strans <= '1';
        when '0' =>
          next_state <= st1;
          strans <= '0';
        when others => next_state <= st0;
      end case;
    when st2 =>
      case sa1 is
        when '1' =>
          next_state <= st3;
          strans <= '1';
        when '0' =>
          next_state <= st2;
          strans <= '0';
        when others => next_state <= st0;
      end case;
    when st3 =>
      case sa3 is
        when '1' =>
          next_state <= st4;
          strans <= '1';
        when '0' =>
          next_state <= st3;
          strans <= '0';
        when others => next_state <= st0;
      end case;
    when st4 =>
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 42/81  
Filename: ASTCDetailedDesign

```
case sa2 is
    when '1' =>
        next_state <= st5;
        strans <= '1';
    when '0' =>
        next_state <= st4;
        strans <= '0';
    when others => next_state <= st0;
end case;
when st5 =>
    case sa2 is
        when '1' =>
            next_state <= st6;
            strans <= '1';
        when '0' =>
            next_state <= st5;
            strans <= '0';
        when others => next_state <= st0;
    end case;
when st6 =>
    case sa2 is
        when '1' =>
            next_state <= st7;
            strans <= '1';
        when '0' =>
            next_state <= st6;
            strans <= '0';
        when others => next_state <= st0;
    end case;
when st7 =>
    case sa2 is
        when '1' =>
            next_state <= st8;
            strans <= '1';
        when '0' =>
            next_state <= st7;
            strans <= '0';
        when others => next_state <= st0;
    end case;
when st8 =>
    case sa4 is
        when '1' =>
            next_state <= st9;
            strans <= '1';
        when '0' =>
            next_state <= st8;
            strans <= '0';
        when others => next_state <= st0;
    end case;
when st9 =>
    case sa5 is
        when '1' =>
            next_state <= st10;
            strans <= '1';
        when '0' =>
            next_state <= st9;
            strans <= '0';
        when others => next_state <= st0;
    end case;
```

```

when st10 =>
  case sa2 is
    when '1' =>
      next_state <= st11;
      strans <= '1';
    when '0' =>
      next_state <= st10;
      strans <= '0';
    when others => next_state <= st0;
  end case;
when st11 =>
  case sa2 is
    when '1' =>
      next_state <= st12;
      strans <= '1';
    when '0' =>
      next_state <= st11;
      strans <= '0';
    when others => next_state <= st0;
  end case;
when st12 =>
  case sa2 is
    when '1' =>
      next_state <= st13;
      strans <= '1';
    when '0' =>
      next_state <= st12;
      strans <= '0';
    when others => next_state <= st0;
  end case;
when st13 =>
  case sa2 is
    when '0' =>
      next_state <= st13;
      strans <= '0';
    when '1' =>
      case sa7 is
        when '0' =>
          next_state <= st4;
          strans <= '1';
        when '1' =>
          next_state <= st14;
          strans <= '1';
        when others => next_state <= st0;
      end case;
      when others => next_state <= st0;
    end case;
when st14 =>
  case sa2 is
    when '1' =>
      next_state <= st15;
      strans <= '1';
    when '0' =>
      next_state <= st14;
      strans <= '0';
    when others => next_state <= st0;
  end case;
when st15 =>
  case sa4 is

```

```

      when '1' =>
          next_state <= st16;
          strans <= '1';
      when '0' =>
          next_state <= st15;
          strans <= '0';
      when others => next_state <= st0;
    end case;
when st16 =>
  case sa2 is
    when '1' =>
        next_state <= st17;
        strans <= '1';
    when '0' =>
        next_state <= st16;
        strans <= '0';
    when others => next_state <= st0;
  end case;
when st17 =>
  case sa6 is
    when '1' =>
        next_state <= st18;
        strans <= '1';
    when '0' =>
        next_state <= st17;
        strans <= '0';
    when others => next_state <= st0;
  end case;
when st18 =>
  case sa2 is
    when '1' =>
        next_state <= st0;
        strans <= '1';
    when '0' =>
        next_state <= st18;
        strans <= '0';
    when others => next_state <= st0;
  end case;
end case;
end process;

-- output definition using pres_state only

process (pres_state)
begin
  case pres_state is
    when st0    => state <= "00000000000000000001";
    when st1    => state <= "000000000000000000010";
    when st2    => state <= "0000000000000000000100";
    when st3    => state <= "00000000000000000001000";
    when st4    => state <= "000000000000000000010000";
    when st5    => state <= "0000000000000000000100000";
    when st6    => state <= "00000000000000000001000000";
    when st7    => state <= "000000000000000000010000000";
    when st8    => state <= "0000000000000000000100000000";
    when st9    => state <= "00000000000000000001000000000";
    when st10   => state <= "000000000000000000010000000000";
  end case;

```

```

when st11  => state <= "00000001000000000000";
when st12  => state <= "00000010000000000000";
when st13  => state <= "00000100000000000000";
when st14  => state <= "00001000000000000000";
when st15  => state <= "00010000000000000000";
when st16  => state <= "00100000000000000000";
when st17  => state <= "01000000000000000000";
when st18  => state <= "10000000000000000000";

end case;
end process;

process (ck, rst, pres_state)
begin
  if rst = '0' then
    ck3 <= '0';
  elsif (ck'event and ck = '1') then
    if pres_state = st1 then
      ck3 <= '1';
    else
      ck3 <= '0';
    end if;
  end if;
end process;

process (ck3, rst)
begin
  if rst = '0' then
    h3 <= (others => '0');
  elsif (ck3'event and ck3 = '1') then
    h3 <= h3 + 1;
  end if;
end process;

h <= std_logic_vector (h3);

process (pres_state, startq, sa2q)
begin
  if pres_state = st17 then
    cl fsm <= '0';
  else
    cl fsm <= startq and sa2q;
  end if;
end process;

saux <= stransq or cl fsm;
p10 <= saux;

end arc_ccd_fsm;

```

#### 4.1.8. decod.vhd

```

-- decod.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 46/81  
Filename: ASTCDetailedDesign

```
use ieee.std_logic_unsigned.all;

entity decod is
    generic (wid: integer);
    port ( a          : in  std_logic_vector (wid downto 0);
           n          : in  integer;
           dec        : out std_logic );
end decod;

architecture arc_decod of decod is

begin

process (a, n)
begin
    if conv_integer(a) = n then
        dec <= '1';
    else dec <= '0';
    end if;
end process;

end arc_decod;
```

### 4.1.9. err\_complete.vhd

```
-- err_complete.vhd

library ieee;
use ieee.std_logic_1164.all;

entity err_complete is
    port ( syncerr      : in  std_logic;
           latchuperr   : in  std_logic;
           parityerr   : in  std_logic;
           st          : in  std_logic_vector (18 downto 0);
           err         : out std_logic_vector (13 downto 0) );
end err_complete;

architecture arc_err_complete of err_complete is

begin

process (latchuperr, st)
begin
    if (st(18) = '1') then
        err(0) <= '0';
    elsif (latchuperr'event and latchuperr = '0') then
        err(0) <= '1';
    end if;
end process;

err(1) <= parityerr;
err(2) <= syncerr;
err(3) <= '0';
err(4) <= '0';
```

```

err(5) <= '0';
err(6) <= '0';
err(7) <= '0';
err(8) <= '0';
err(9) <= '0';
err(10) <= '1';
err(11) <= '0';
err(12) <= '0';
err(13) <= '0';

end arc_err_complete;

```

#### 4.1.10. rx\_complete.vhd

```

-- rx_complete.vhd

library ieee;
use ieee.std_logic_1164.all;

entity rx_complete is
  port ( din      : in    std_logic;
         sin      : in    std_logic;
         ck       : in    std_logic;
         rst      : in    std_logic;
         parityerr : out   std_logic;
         csetup   : out   std_logic_vector (13 downto 0);
         start    : out   std_logic;
         hvref    : out   std_logic;
         ckric   : out   std_logic;
         st       : in    std_logic_vector (18 downto 0);
         f        : in    std_logic_vector (6 downto 0) );
end rx_complete;

architecture arc_rx_complete of rx_complete is

component rx_read
  port ( rst      : in    std_logic;
         din     : in    std_logic;
         ck      : in    std_logic;
         syncdata : in    std_logic;
         dric   : out   std_logic_vector (15 downto 0);
         st      : in    std_logic_vector (18 downto 0) );
end component;

component rx_ck_rec
  port ( din      : in    std_logic;
         sin      : in    std_logic;
         ck       : in    std_logic;
         rst      : in    std_logic;
         ckric   : out   std_logic;
         syncr   : out   std_logic );
end component;

component rx_sync_gen
  port ( ck      : in    std_logic;
         rst     : in    std_logic;

```

```

      syncr      : in      std_logic;
      syncdata   : out     std_logic );
end component;

component rx_data_rec
  port ( dric      : in      std_logic_vector (15 downto 0);
         syncdata  : in      std_logic;
         rst       : in      std_logic;
         parityerr : out     std_logic;
         csetup    : out     std_logic_vector (13 downto 0);
         start     : out     std_logic;
         hvref     : out     std_logic;
         st        : in      std_logic_vector (18 downto 0);
         f         : in      std_logic_vector (6 downto 0) );
end component;

signal      syncr      : std_logic;
signal      syncdata   : std_logic;
signal      dric       : std_logic_vector (15 downto 0);

begin

inst1 : rx_read
  port map ( rst=>rst, din=>din, ck=>ck, syncdata=>syncdata,
             dric(15 downto 0)=>dric(15 downto 0),
             st(18 downto 0)=>st(18 downto 0) );
inst2 : rx_ck_rec
  port map ( din=>din, sin=>sin, ck=>ck, rst=>rst,
             ckric=>ckric, syncr=>syncr );
inst3 : rx_sync_gen
  port map ( ck=>ck, rst=>rst, syncr=>syncr,
             syncdata=>syncdata );
inst4 : rx_data_rec
  port map ( dric(15 downto 0)=>dric(15 downto 0),
             syncdata=>syncdata, rst=>rst,
             parityerr=>parityerr,
             csetup(13 downto 0)=>csetup(13 downto 0),
             start=>start, hvref=>hvref,
             st(18 downto 0)=>st(18 downto 0),
             f(6 downto 0)=>f(6 downto 0) );

end arc_rx_complete;

```

#### 4.1.11. rx\_read.vhd

```
-- rx_read.vhd

library ieee;
use ieee.std_logic_1164.all;

entity rx_read is
  port ( rst      : in      std_logic;
         din     : in      std_logic;
         ck      : in      std_logic;
         syncdata : in      std_logic;
         dric    : out     std_logic_vector (15 downto 0);
         st      : in      std_logic_vector (18 downto 0) );

```

```

end rx_read;

architecture arc_rx_read of rx_read is

  signal      q1          : std_logic_vector (7 downto 0);
  signal      q2          : std_logic_vector (7 downto 0);
  signal      rst0        : std_logic;

begin

  process (rst, ck, din)
  begin
    if rst = '0' then
      q1 <= (others => '0');
    elsif (ck'event and ck = '0') then
      q1 <= q1(6 downto 0) & din;
    end if;
  end process;

  process (rst, ck)
  begin
    if rst = '0' then
      q2 <= (others => '0');
    elsif (ck'event and ck = '1') then
      q2 <= q2(6 downto 0) & din;
    end if;
  end process;

  process (rst0, syncdata)
  begin
    if rst0 = '0' then
      dric <= (others => '0');
    elsif (syncdata'event and syncdata = '1') then
      for i in 0 to 7 loop
        dric(2*i) <= q2(7-i);
        dric(2*i+1) <= q1(7-i);
      end loop;
    end if;
  end process;

  rst0 <= rst and not st(0);

end arc_rx_read;

```

#### 4.1.12. rx\_ck\_rec.vhd

```

-- rx_ck_rec.vhd

library ieee;
use ieee.std_logic_1164.all;

entity rx_ck_rec is
  port ( din      : in  std_logic;
         sin      : in  std_logic;
         ck       : in  std_logic;
         rst     : in  std_logic;

```

```

      ckric      : out    std_logic;
      syncr      : out    std_logic );
end rx_ck_rec;

architecture arc_rx_ck_rec of rx_ck_rec is

  component my_and7
    port( data        : in std_logic_vector(6 downto 0);
          result      : out std_logic) ;
  end component;

  component my_and8
    port( data        : in std_logic_vector(7 downto 0);
          result      : out std_logic) ;
  end component;

  signal      a1      : std_logic;
  signal      a2      : std_logic;
  signal      q1      : std_logic_vector (6 downto 0);
  signal      q17     : std_logic;
  signal      q2      : std_logic_vector (7 downto 0);

begin

  inst2 : my_and7 port map ( data => q1, result => a1);
  inst3 : my_and8 port map ( data => q2, result => a2);

  process(ck, rst)
  begin
    if rst = '0' then
      q1 <= (others => '0');
    elsif (ck'event and ck = '1') then
      q1 <= q1(5 downto 0) & din;
    end if;
  end process;

  process(ck, rst)
  begin
    if rst = '0' then
      q17 <= '1';
    elsif (ck'event and ck = '1') then
      q17 <= not q1(6);
    end if;
  end process;

  process(ck, rst)
  begin
    if rst = '0' then
      q2 <= (others => '0');
    elsif (ck'event and ck = '0') then
      q2 <= q2(6 downto 0) & din;
    end if;
  end process;

  ckric <= din xor sin;
  syncr <= a1 and a2 and q17;

end arc_rx_ck_rec;

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 51/81  
Filename: ASTCDetailedDesign

### 4.1.13. my\_and7.vhd

```
-- Version: 6.1 6.1.1.24

library ieee;
use ieee.std_logic_1164.all;
library a54SXA;

entity my_and7 is
    port( Data : in std_logic_vector(6 downto 0); Result : out
          std_logic) ;
end my_and7;

architecture DEF_ARCH of my_and7 is

    component AND4
        port(A, B, C, D : in std_logic; Y : out std_logic) ;
    end component;

    signal AND4_0_Y : std_logic ;
begin

    AND4_0 : AND4
        port map(A => Data(3), B => Data(4), C => Data(5), D =>
                 Data(6), Y => AND4_0_Y);
    AND4_Result : AND4
        port map(A => AND4_0_Y, B => Data(2), C => Data(1), D =>
                 Data(0), Y => Result);
end DEF_ARCH;
```

### 4.1.14. my\_and8.vhd

```
-- Version: 6.1 6.1.1.24

library ieee;
use ieee.std_logic_1164.all;
library a54SXA;

entity my_and8 is
    port( Data : in std_logic_vector(7 downto 0); Result : out
          std_logic) ;
end my_and8;

architecture DEF_ARCH of my_and8 is

    component AND2
        port(A, B : in std_logic; Y : out std_logic) ;
    end component;

    component AND4
```

```

    port(A, B, C, D : in std_logic;  Y : out std_logic) ;
end component;

signal AND4_0_Y, AND2_0_Y : std_logic ;
begin

AND2_0 : AND2
  port map(A => Data(2), B => Data(3), Y => AND2_0_Y);
AND4_0 : AND4
  port map(A => Data(4), B => Data(5), C => Data(6), D =>
           Data(7), Y => AND4_0_Y);
AND4_Result : AND4
  port map(A => AND4_0_Y, B => AND2_0_Y, C => Data(1), D =>
           Data(0), Y => Result);
end DEF_ARCH;

```

#### 4.1.15. rx\_sync\_gen.vhd

```

-- rx_sync_gen.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity rx_sync_gen is
  port ( ck          : in    std_logic;
         rst         : in    std_logic;
         syncr      : in    std_logic;
         syncdata   : out   std_logic );
end rx_sync_gen;

architecture arc_rx_sync_gen of rx_sync_gen is

  signal     a1      : std_logic;
  signal     q1      : std_logic;
  signal     q       : std_logic_vector (2 downto 0);

begin

process(rst, ck, syncr)
begin
  if rst = '0' then
    q <= (others => '0');
  elsif (ck'event and ck = '1') then
    if syncr = '1' then
      q <= (others => '0');
    else q <= q + '1';
    end if;
  end if;
end process;

process (rst, ck)
begin
  if rst = '0' then
    q1 <= '0';
  end if;
end process;

```

```

      elsif (ck'event and ck = '0') then
        q1 <= a1;
      end if;
    end process;

    a1 <= q(0) and q(1) and q(2);
    syncdata <= a1 and q1;

end arc_rx_sync_gen;

```

#### 4.1.16. rx\_data\_rec.vhd

```

-- rx_data_rec.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity rx_data_rec is
  port ( dric      : in  std_logic_vector (15 downto 0);
         syncdata   : in  std_logic;
         rst        : in  std_logic;
         parityerr  : out std_logic;
         csetup     : out std_logic_vector (13 downto 0);
         start      : out std_logic;
         hvref      : out std_logic;
         st         : in  std_logic_vector (18 downto 0);
         f          : in  std_logic_vector (6 downto 0) );
end rx_data_rec;

architecture arc_rx_data_rec of rx_data_rec is

component my_xor14
  port( data      : in std_logic_vector(13 downto 0);
        result    : out std_logic) ;
end component;

signal a0           : std_logic;
signal parityerrs  : std_logic;
signal q            : std_logic_vector (14 downto 0);
signal csetup      : std_logic_vector (13 downto 0);
signal a3           : std_logic;
signal ckdout      : std_logic;
signal a1           : std_logic;
signal a2           : std_logic;
signal a4           : std_logic;
signal r            : std_logic_vector (2 downto 0);
signal rst0         : std_logic;
signal syncdata1    : std_logic;
signal syncdata2    : std_logic;

begin

  inst1 : my_xor14 port map ( data => q(14 downto 1), result => a0);

  process (rst, syncdata1)

```

```

begin
  if rst = '0' then
    q <= (others => '0');
  elsif (syncdata1'event and syncdata1 = '1') then
    q <= dric(15 downto 1);
  end if;
end process;

process (rst, syncdata2)
begin
  if rst = '0' then
    csetups <= (others => '0');
    parityerrs <= '1';
    a4 <= '1';
  elsif (syncdata2'event and syncdata2 = '1') then
    csetups <= q(14 downto 1);
    parityerrs <= not(a0 xor dric(0) xor dric(1));
    a4 <= q(0);
  end if;
end process;

process (rst0, ckdout, a3)
begin
  if rst0 = '0' then
    start <= '0';
  elsif (ckdout'event and ckdout = '1') then
    case a3 is
      when '1' =>
        start <= csetups(6);
      when others =>
    end case;
  end if;
end process;

process (rst, ckdout, a3)
begin
  if rst = '0' then
    r <= (others => '0');
  elsif (ckdout'event and ckdout = '1') then
    case a3 is
      when '1' =>
        r <= csetups(2 downto 0);
      when others =>
    end case;
  end if;
end process;

process (ckdout, rst)
begin
  if rst = '0' then
    parityerr <= '0';
    csetup <= (others => '0');
  elsif (ckdout'event and ckdout = '1') then
    parityerr <= parityerrs;
    csetup <= csetups;
  end if;
end process;

process (r, f)

```

```

begin
  case r is
    when "111" =>
      hvref <= f(0);
    when "110" =>
      hvref <= f(1);
    when "101" =>
      hvref <= f(2);
    when "100" =>
      hvref <= f(3);
    when "011" =>
      hvref <= f(4);
    when "010" =>
      hvref <= f(5);
    when "001" =>
      hvref <= f(6);
    when "000" =>
      hvref <= '0';
    when others => hvref <= '0';
  end case;
end process;

rst0 <= rst and not st(0);

a1 <= csetups(10) and not csetups(11) and not csetups(12) and not
csetups(13);
a2 <= not csetups(7) and not csetups(8) and not csetups(9);
a3 <= not parityerrs and a1 and a2;

syncdata1 <= syncdata;
syncdata2 <= not syncdata;

ckdout <= not a4 and syncdata;

end arc_rx_data_rec;

```

#### 4.1.17. my\_xor14.vhd

```

-- Version: 6.1 6.1.1.24

library ieee;
use ieee.std_logic_1164.all;
library a54SXA;

entity my_xor14 is
  port( Data : in std_logic_vector(13 downto 0); Result : out
        std_logic) ;
end my_xor14;

architecture DEF_ARCH of my_xor14 is

  component CM8INV
    port(A : in std_logic; Y : out std_logic) ;
  end component;

```

```

component CM8F
  port(D0, D1, D2, D3, S00, S01, S10, S11 : in std_logic;
       Y, FY : out std_logic) ;
end component;

component VCC
  port( Y : out std_logic);
end component;

component GND
  port( Y : out std_logic);
end component;

signal Data_15_net, Data_16_net, Data_17_net, Data_14_net,
   Data_18_net, Data_19_net, CM8INV_1_Y, CM8INV_5_Y,
   CM8INV_2_Y, CM8INV_3_Y, CM8INV_0_Y, CM8INV_4_Y, VCC_1_net,
   GND_1_net : std_logic ;
begin

  VCC_2_net : VCC port map(Y => VCC_1_net);
  GND_2_net : GND port map(Y => GND_1_net);
  CM8INV_2 : CM8INV
    port map(A => Data_17_net, Y => CM8INV_2_Y);
  CM8F_Data_16_inst : CM8F
    port map(D0 => CM8INV_1_Y, D1 => Data(8), D2 => Data(8),
             D3 => CM8INV_1_Y, S00 => VCC_1_net, S01 => Data(6),
             S10 => Data(7), S11 => GND_1_net, Y => OPEN , FY =>
             Data_16_net);
  CM8INV_5 : CM8INV
    port map(A => Data(5), Y => CM8INV_5_Y);
  CM8F_Data_15_inst : CM8F
    port map(D0 => CM8INV_5_Y, D1 => Data(5), D2 => Data(5),
             D3 => CM8INV_5_Y, S00 => VCC_1_net, S01 => Data(3),
             S10 => Data(4), S11 => GND_1_net, Y => OPEN , FY =>
             Data_15_net);
  CM8INV_4 : CM8INV
    port map(A => Data(11), Y => CM8INV_4_Y);
  CM8F_Data_18_inst : CM8F
    port map(D0 => CM8INV_3_Y, D1 => Data_14_net, D2 =>
             Data_14_net, D3 => CM8INV_3_Y, S00 => VCC_1_net, S01 =>
             Data(12), S10 => Data(13), S11 => GND_1_net, Y => OPEN ,
             FY => Data_18_net);
  CM8F_Data_17_inst : CM8F
    port map(D0 => CM8INV_4_Y, D1 => Data(11), D2 => Data(11),
             D3 => CM8INV_4_Y, S00 => VCC_1_net, S01 => Data(9),
             S10 => Data(10), S11 => GND_1_net, Y => OPEN , FY =>
             Data_17_net);
  CM8F_Data_14_inst : CM8F
    port map(D0 => Data(2), D1 => CM8INV_0_Y, D2 => CM8INV_0_Y,
             D3 => Data(2), S00 => VCC_1_net, S01 => Data(0), S10 =>
             GND_1_net, S11 => Data(1), Y => OPEN , FY => Data_14_net);
  CM8F_Data_19_inst : CM8F
    port map(D0 => CM8INV_2_Y, D1 => Data_17_net, D2 =>
             Data_17_net, D3 => CM8INV_2_Y, S00 => VCC_1_net, S01 =>
             Data_15_net, S10 => Data_16_net, S11 => GND_1_net, Y =>
             OPEN , FY => Data_19_net);
  CM8INV_0 : CM8INV
    port map(A => Data(2), Y => CM8INV_0_Y);

```

```

CM8INV_1 : CM8INV
  port map(A => Data(8), Y => CM8INV_1_Y);
CM8INV_3 : CM8INV
  port map(A => Data_14_net, Y => CM8INV_3_Y);
CM8F_Result : CM8F
  port map(D0 => VCC_1_net, D1 => GND_1_net, D2 => GND_1_net,
            D3 => VCC_1_net, S00 => VCC_1_net, S01 => Data_18_net,
            S10 => Data_19_net, S11 => GND_1_net, Y => OPEN , FY =>
            Result);
end DEF_ARCH;

```

#### 4.1.18. hk\_complete.vhd

```

-- hk_complete.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity hk_complete is
  port ( qs          : out    std_logic_vector(3 downto 0) ;
         rst         : in     std_logic;
         hkdout      : in     std_logic;
         sstrb       : in     std_logic;
         st          : in     std_logic_vector (18 downto 0);
         ck          : in     std_logic;
         hk          : out    std_logic_vector (13 downto 0);
         hkdin      : out    std_logic;
         sclk        : out    std_logic;
         cs          : out    std_logic;
         h           : in     std_logic_vector (2 downto 0) );
end hk_complete;

architecture arc_hk_complete of hk_complete is

component clkint
  port ( a          : in     std_logic;
         y          : out    std_logic );
end component;

constant maxcount      : unsigned (3 downto 0):="1011";

signal      ckint      : std_logic;
signal      sel        : std_logic;
signal      q          : std_logic_vector (9 downto 0);
signal      q1         : unsigned (3 downto 0);
signal      q4         : std_logic_vector (7 downto 0);

begin

inst0: clkint port map (a=>ck, y=>ckint);

process (ckint, sel, rst)
begin
  if rst = '0' then

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 58/81  
Filename: ASTCDetailedDesign

```
q <= (others => '0');
elsif (ckint'event and ckint = '1') then
  if (sel = '0') then
    q <= q(8 downto 0) & hkdout;
  end if;
end if;
end process;

process(ckint, sstrb, sel, rst)
begin
  if rst = '0' or sstrb = '1' then
    q1 <= (others => '0');
  elsif (ckint'event and ckint = '0') then
    if sel = '0' then
      q1 <= q1 + 1;
    end if;
  end if;
end process;

process (q1)
begin
  if q1 = maxcount then
    sel <= '1';
  else
    sel <= '0';
  end if;
end process;

process (ckint, st(3), rst)
begin
  if rst = '0' then
    q4 <= (others => '0');
  elsif (ckint'event and ckint = '1') then
    if (st(3) = '0') then
      q4 <= '1' & h(0) & h(2) & h(1) & "1111";
    else
      q4 <= q4(6 downto 0) & '0';
    end if;
  end if;
end process;

process (ckint, st(3))
begin
  if (ckint'event and ckint = '1') then
    if (st(3) = '0') then
      hkdin <= '0';
    else hkdin <= q4(7);
    end if;
  end if;
end process;

hk <= h(2) & h(1) & h(0) & '0' & q;
sclk <= not ckint;
cs <= '0';

qs <= std_logic_vector(q1);

end arc_hk_complete;
```

#### 4.1.19. tx\_complete.vhd

```
-- tx_complete.vhd

library ieee;
use ieee.std_logic_1164.all;

entity tx_complete is
  port ( rst      : in    std_logic;
         datain   : in    std_logic_vector (13 downto 0);
         hk       : in    std_logic_vector (13 downto 0);
         err      : in    std_logic_vector (13 downto 0);
         csetup   : in    std_logic_vector (13 downto 0);
         fd       : in    std_logic;
         st       : in    std_logic_vector (18 downto 0);
         ck       : in    std_logic;
         ck40    : in    std_logic;
         dout    : out   std_logic;
         sout    : out   std_logic );
end tx_complete;

architecture arc_tx_complete of tx_complete is

component tx_mux
  port ( rst      : in    std_logic;
         din     : in    std_logic_vector (13 downto 0);
         hk      : in    std_logic_vector (13 downto 0);
         err     : in    std_logic_vector (13 downto 0);
         csetup  : in    std_logic_vector (13 downto 0);
         st      : in    std_logic_vector (18 downto 0);
         ck      : in    std_logic;
         fd      : in    std_logic;
         data    : out   std_logic_vector (13 downto 0);
         cdn    : out   std_logic );
end component;

component tx_count
  port ( rst      : in    std_logic;
         st       : in    std_logic_vector (18 downto 0);
         ck       : in    std_logic;
         syncck  : out   std_logic;
         ctrl    : out   std_logic_vector (13 downto 0) );
end component;

component tx_data_gen
  port ( rst      : in    std_logic;
         data    : in    std_logic_vector (13 downto 0);
         ctrl   : in    std_logic_vector (13 downto 0);
         cdn    : in    std_logic;
         syncck : in    std_logic;
         ck     : in    std_logic;
         d      : buffer std_logic_vector (15 downto 0);
         decs   : out   std_logic );
end component;
```

```

component tx_ds_gen
  port ( rst          : in      std_logic;
         d            : in      std_logic_vector (15 downto 0);
         decs         : in      std_logic;
         ck           : in      std_logic;
         ck40         : in      std_logic;
         dout         : out     std_logic;
         sout         : out     std_logic );
end component;

signal    cdn      : std_logic;
signal    syncck   : std_logic;
signal    decs     : std_logic;
signal    data     : std_logic_vector (13 downto 0);
signal    ctrl     : std_logic_vector (13 downto 0);
signal    d        : std_logic_vector (15 downto 0);

begin

inst1 : tx_mux
  port map ( rst=>rst,
             din(13 downto 0)=>datain(13 downto 0),
             hk(13 downto 0)=>hk(13 downto 0),
             err(13 downto 0)=>err(13 downto 0),
             csetup(13 downto 0)=>csetup(13 downto 0),
             st(18 downto 0)=>st(18 downto 0), ck=>ck, fd=>fd,
             data(13 downto 0)=>data(13 downto 0), cdn=>cdn );
inst2 : tx_count
  port map ( rst=>rst,
             st(18 downto 0)=>st(18 downto 0), ck=>ck, syncck=>syncck,
             ctrl(13 downto 0)=>ctrl(13 downto 0) );
inst3 : tx_data_gen
  port map ( rst=>rst,
             data(13 downto 0)=>data(13 downto 0),
             ctrl(13 downto 0)=>ctrl(13 downto 0),
             cdn=>cdn, syncck=>syncck, ck=>ck,
             d(15 downto 0)=>d(15 downto 0),
             decs=>decs );
inst4 : tx_ds_gen
  port map ( rst=>rst,
             d(15 downto 0)=>d(15 downto 0), decs=>decs, ck=>ck,
             ck40=>ck40, dout=>dout, sout=>sout );

end arc_tx_complete;

```

#### 4.1.20. tx\_mux.vhd

```

-- tx_mux.vhd

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity tx_mux is
  port ( rst          : in      std_logic;
         din         : in      std logic vector (13 downto 0);

```

```

      hk      : in      std_logic_vector (13 downto 0);
      err     : in      std_logic_vector (13 downto 0);
      csetup  : in      std_logic_vector (13 downto 0);
      st      : in      std_logic_vector (18 downto 0);
      ck      : in      std_logic;
      fd      : in      std_logic;
      data    : out     std_logic_vector (13 downto 0);
      cdn    : out     std_logic );
end tx_mux;

architecture arc_tx_mux of tx_mux is

component my_mux4
  port( data0_port : in std_logic_vector(13 downto 0);
        data1_port : in std_logic_vector(13 downto 0);
        data2_port : in std_logic_vector(13 downto 0);
        data3_port : in std_logic_vector(13 downto 0);
        sel0      : in std_logic;
        sel1      : in std_logic;
        result    : out std_logic_vector(13 downto 0) );
end component;

signal s0      : std_logic;
signal s1      : std_logic;
signal a       : std_logic;
signal d       : std_logic_vector (13 downto 0);
signal q       : std_logic_vector (5 downto 0);

begin

inst1 : my_mux4
  port map ( data0_port(13 downto 0)=>d(13 downto 0),
             data1_port(13 downto 0)=>hk(13 downto 0),
             data2_port(13 downto 0)=>err(13 downto 0),
             data3_port(13 downto 0)=>csetup(13 downto 0),
             sel0=>s0, sel1=>s1,
             result(13 downto 0)=>data(13 downto 0) );

process (din, st(15))
begin
  for i in 0 to 13 loop
    d(i) <= din(i) and not st(15);
  end loop;
end process;

process(ck, st(15), rst)
begin
  if rst = '0' then
    q <= (others => '0');
  elsif (ck'event and ck = '1') then
    if st(15) = '0' then
      q <= (others => '0');
    else q <= q + '1';
    end if;
  end if;
end process;

a <= not q(5) and not q(4) and not q(3) and not q(2);
s0 <= a and q(0);

```

```

s1 <= a and q(1);
cdn <= not ( fd or (not q(3) and ((q(2) and not q(1) and not q(0)) or
                           ((not q(2) and (q(1) or q(0)))))) ) ;

end arc_tx_mux;

```

#### 4.1.21. my\_mux4.vhd

```

-- Version: 6.1 6.1.1.24

library ieee;
use ieee.std_logic_1164.all;
library a54SXA;

entity my_mux4 is
  port( Data0_port : in std_logic_vector(13 downto 0);
        Data1_port : in std_logic_vector(13 downto 0); Data2_port :
        in std_logic_vector(13 downto 0); Data3_port : in
        std_logic_vector(13 downto 0); Sel0, Sel1 : in std_logic;
        Result : out std_logic_vector(13 downto 0)) ;
end my_mux4;

architecture DEF_ARCH of my_mux4 is

  component MX4
    port(D0, D1, D2, D3, S0, S1 : in std_logic; Y : out
         std_logic) ;
  end component;

  component BUFF
    port(A : in std_logic; Y : out std_logic) ;
  end component;

  signal SelAux_0_0_net, SelAux_0_5_net, SelAux_0_10_net,
         SelAux_1_0_net, SelAux_1_5_net, SelAux_1_10_net : std_logic ;
begin

  MX4_Result_0_inst : MX4
    port map(D0 => Data0_port(0), D1 => Data1_port(0), D2 =>
              Data2_port(0), D3 => Data3_port(0), S0 => SelAux_0_0_net,
              S1 => SelAux_1_0_net, Y => Result(0));
  BUFF_SelAux_1_5_inst : BUFF
    port map(A => Sel1, Y => SelAux_1_5_net);
  MX4_Result_2_inst : MX4
    port map(D0 => Data0_port(2), D1 => Data1_port(2), D2 =>
              Data2_port(2), D3 => Data3_port(2), S0 => SelAux_0_0_net,
              S1 => SelAux_1_0_net, Y => Result(2));
  MX4_Result_12_inst : MX4
    port map(D0 => Data0_port(12), D1 => Data1_port(12), D2 =>
              Data2_port(12), D3 => Data3_port(12), S0 =>
              SelAux_0_10_net, S1 => SelAux_1_10_net, Y => Result(12));
  MX4_Result_5_inst : MX4
    port map(D0 => Data0_port(5), D1 => Data1_port(5), D2 =>
              Data2_port(5), D3 => Data3_port(5), S0 => SelAux_0_5_net,
              S1 => SelAux_1_5_net, Y => Result(5));
  MX4_Result_11_inst : MX4

```

```

port map(D0 => Data0_port(11), D1 => Data1_port(11), D2 =>
         Data2_port(11), D3 => Data3_port(11), S0 =>
         SelAux_0_10_net, S1 => SelAux_1_10_net, Y => Result(11));
MX4_Result_1_inst : MX4
port map(D0 => Data0_port(1), D1 => Data1_port(1), D2 =>
         Data2_port(1), D3 => Data3_port(1), S0 => SelAux_0_0_net,
         S1 => SelAux_1_0_net, Y => Result(1));
BUFF_SelAux_1_10_inst : BUFF
port map(A => Sel1, Y => SelAux_1_10_net);
BUFF_SelAux_1_0_inst : BUFF
port map(A => Sel1, Y => SelAux_1_0_net);
MX4_Result_7_inst : MX4
port map(D0 => Data0_port(7), D1 => Data1_port(7), D2 =>
         Data2_port(7), D3 => Data3_port(7), S0 => SelAux_0_5_net,
         S1 => SelAux_1_5_net, Y => Result(7));
MX4_Result_4_inst : MX4
port map(D0 => Data0_port(4), D1 => Data1_port(4), D2 =>
         Data2_port(4), D3 => Data3_port(4), S0 => SelAux_0_0_net,
         S1 => SelAux_1_0_net, Y => Result(4));
BUFF_SelAux_0_10_inst : BUFF
port map(A => Sel0, Y => SelAux_0_10_net);
MX4_Result_3_inst : MX4
port map(D0 => Data0_port(3), D1 => Data1_port(3), D2 =>
         Data2_port(3), D3 => Data3_port(3), S0 => SelAux_0_0_net,
         S1 => SelAux_1_0_net, Y => Result(3));
BUFF_SelAux_0_0_inst : BUFF
port map(A => Sel0, Y => SelAux_0_0_net);
MX4_Result_6_inst : MX4
port map(D0 => Data0_port(6), D1 => Data1_port(6), D2 =>
         Data2_port(6), D3 => Data3_port(6), S0 => SelAux_0_5_net,
         S1 => SelAux_1_5_net, Y => Result(6));
MX4_Result_9_inst : MX4
port map(D0 => Data0_port(9), D1 => Data1_port(9), D2 =>
         Data2_port(9), D3 => Data3_port(9), S0 => SelAux_0_5_net,
         S1 => SelAux_1_5_net, Y => Result(9));
MX4_Result_13_inst : MX4
port map(D0 => Data0_port(13), D1 => Data1_port(13), D2 =>
         Data2_port(13), D3 => Data3_port(13), S0 =>
         SelAux_0_10_net, S1 => SelAux_1_10_net, Y => Result(13));
BUFF_SelAux_0_5_inst : BUFF
port map(A => Sel0, Y => SelAux_0_5_net);
MX4_Result_10_inst : MX4
port map(D0 => Data0_port(10), D1 => Data1_port(10), D2 =>
         Data2_port(10), D3 => Data3_port(10), S0 =>
         SelAux_0_10_net, S1 => SelAux_1_10_net, Y => Result(10));
MX4_Result_8_inst : MX4
port map(D0 => Data0_port(8), D1 => Data1_port(8), D2 =>
         Data2_port(8), D3 => Data3_port(8), S0 => SelAux_0_5_net,
         S1 => SelAux_1_5_net, Y => Result(8));
end DEF_ARCH;

```

#### 4.1.22. tx\_count.vhd

```
-- tx_count.vhd

library ieee;
use ieee.std_logic_1164.all;
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 64/81  
Filename: ASTCDetailedDesign

```
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity tx_count is
    port ( rst          : in      std_logic;
           st           : in      std_logic_vector (18 downto 0);
           ck           : in      std_logic;
           syncck       : out     std_logic;
           ctrl         : out     std_logic_vector (13 downto 0) );
end tx_count;

architecture arc_tx_count of tx_count is

    signal      syncw      : std_logic;
    signal      comdata    : std_logic;
    signal      a          : std_logic;
    signal      s          : std_logic;
    signal      q          : std_logic_vector (3 downto 0);

begin

    process(ck, s, rst)
    begin
        if rst = '0' then
            q <= (others => '0');
        elsif (ck'event and ck = '1') then
            if s = '0' then
                q <= (others => '0');
            else q <= q + '1';
            end if;
        end if;
    end process;

    process (syncw, a, comdata, st)
    begin
        if syncw = '1' then
            ctrl(0)<= '1';
            ctrl(1)<= '1';
            ctrl(2)<= '1';
            ctrl(3)<= '1';
            ctrl(4)<= '1';
            ctrl(5)<= '1';
        else
            ctrl(0)<= a;
            ctrl(1)<= a;
            ctrl(2)<= comdata and (st(1) or st(14));
            ctrl(3)<= comdata and st(16);
            ctrl(4)<= comdata and st(5);
            ctrl(5)<= comdata and (st(12) or st(14));
        end if;
    end process;

    s<= st(1) or st(5) or st(12) or st(14) or st(16);
    a<= q(0) and not q(1) and not q(2) and not q(3);
    syncw<= not q(0) and q(1) and not q(2) and not q(3);
    comdata<= q(0) and q(1) and not q(2) and not q(3);
    ctrl(6)<= syncw;
    ctrl(7)<= syncw;
    ctrl(8)<= syncw;
```

```

ctrl(9)<= syncw;
ctrl(10)<= syncw;
ctrl(11)<= syncw;
ctrl(12)<= syncw;
ctrl(13)<= syncw;
syncck <= a;

end arc_tx_count;

```

#### 4.1.23. tx\_data\_gen.vhd

```

-- tx_data_gen.vhd

library ieee;
use ieee.std_logic_1164.all;

entity tx_data_gen is
  port ( rst          : in      std_logic;
         data         : in      std_logic_vector (13 downto 0);
         ctrl         : in      std_logic_vector (13 downto 0);
         cdn          : in      std_logic;
         syncck       : in      std_logic;
         ck           : in      std_logic;
         d            : buffer  std_logic_vector (15 downto 0);
         decs         : out     std_logic );
end tx_data_gen;

architecture arc_tx_data_gen of tx_data_gen is

  component my_xor14
    port( data      : in std_logic_vector(13 downto 0);
          result    : out std_logic) ;
  end component;

  signal a        : std_logic;
  signal parity   : std_logic;
  signal q        : std_logic_vector (13 downto 0);

begin

  inst1 : my_xor14 port map ( data => d(15 downto 2), result => a);

  process (cdn, data, ctrl)
  begin
    case (cdn) is
      when '0' =>
        q<= data;
      when '1' =>
        q<= ctrl;
      when others => q <= "-----";
    end case;
  end process;

  process (ck, rst)
  begin
    if rst = '0' then
      d <= (others => '0');
    end if;
  end process;

```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 66/81  
Filename: ASTCDetailedDesign

```
elsif (ck'event and ck = '0') then
    for i in 2 to 15 loop
        d(i) <= q(i-2);
    end loop;
    d(0) <= parity;
    d(1) <= cdn;
    end if;
end process;

process (ck, rst)
begin
    if rst = '0' then
        decs <= '0';
    elsif (ck'event and ck = '0') then
        decs <= syncck;
    end if;
end process;

parity<= a xnor cdn;

end arc_tx_data_gen;
```

### 4.1.24. my\_xor14.vhd

```
-- Version: 6.1 6.1.1.24

library ieee;
use ieee.std_logic_1164.all;
library a54SXA;

entity my_xor14 is
    port( Data : in std_logic_vector(13 downto 0); Result : out
          std_logic) ;
end my_xor14;

architecture DEF_ARCH of my_xor14 is

    component CM8INV
        port(A : in std_logic; Y : out std_logic) ;
    end component;

    component CM8F
        port(D0, D1, D2, D3, S00, S01, S10, S11 : in std_logic;
             Y, FY : out std_logic) ;
    end component;

    component VCC
        port( Y : out std_logic);
    end component;

    component GND
        port( Y : out std_logic);
    end component;

    signal Data_15_net, Data_16_net, Data_17_net, Data_14_net,
           Data_18_net, Data_19_net, CM8INV_1_Y, CM8INV_5_Y,
```

```

CM8INV_2_Y, CM8INV_3_Y, CM8INV_0_Y, CM8INV_4_Y, VCC_1_net,
GND_1_net : std_logic ;
begin

VCC_2_net : VCC port map(Y => VCC_1_net);
GND_2_net : GND port map(Y => GND_1_net);
CM8INV_2 : CM8INV
  port map(A => Data_17_net, Y => CM8INV_2_Y);
CM8F_Data_16_inst : CM8F
  port map(D0 => CM8INV_1_Y, D1 => Data(8), D2 => Data(8),
    D3 => CM8INV_1_Y, S00 => VCC_1_net, S01 => Data(6),
    S10 => Data(7), S11 => GND_1_net, Y => OPEN , FY =>
    Data_16_net);
CM8INV_5 : CM8INV
  port map(A => Data(5), Y => CM8INV_5_Y);
CM8F_Data_15_inst : CM8F
  port map(D0 => CM8INV_5_Y, D1 => Data(5), D2 => Data(5),
    D3 => CM8INV_5_Y, S00 => VCC_1_net, S01 => Data(3),
    S10 => Data(4), S11 => GND_1_net, Y => OPEN , FY =>
    Data_15_net);
CM8INV_4 : CM8INV
  port map(A => Data(11), Y => CM8INV_4_Y);
CM8F_Data_18_inst : CM8F
  port map(D0 => CM8INV_3_Y, D1 => Data_14_net, D2 =>
    Data_14_net, D3 => CM8INV_3_Y, S00 => VCC_1_net, S01 =>
    Data(12), S10 => Data(13), S11 => GND_1_net, Y => OPEN ,
    FY => Data_18_net);
CM8F_Data_17_inst : CM8F
  port map(D0 => CM8INV_4_Y, D1 => Data(11), D2 => Data(11),
    D3 => CM8INV_4_Y, S00 => VCC_1_net, S01 => Data(9),
    S10 => Data(10), S11 => GND_1_net, Y => OPEN , FY =>
    Data_17_net);
CM8F_Data_14_inst : CM8F
  port map(D0 => Data(2), D1 => CM8INV_0_Y, D2 => CM8INV_0_Y,
    D3 => Data(2), S00 => VCC_1_net, S01 => Data(0), S10 =>
    GND_1_net, S11 => Data(1), Y => OPEN , FY => Data_14_net);
CM8F_Data_19_inst : CM8F
  port map(D0 => CM8INV_2_Y, D1 => Data_17_net, D2 =>
    Data_17_net, D3 => CM8INV_2_Y, S00 => VCC_1_net, S01 =>
    Data_15_net, S10 => Data_16_net, S11 => GND_1_net, Y =>
    OPEN , FY => Data_19_net);
CM8INV_0 : CM8INV
  port map(A => Data(2), Y => CM8INV_0_Y);
CM8INV_1 : CM8INV
  port map(A => Data(8), Y => CM8INV_1_Y);
CM8INV_3 : CM8INV
  port map(A => Data_14_net, Y => CM8INV_3_Y);
CM8F_Result : CM8F
  port map(D0 => VCC_1_net, D1 => GND_1_net, D2 => GND_1_net,
    D3 => VCC_1_net, S00 => VCC_1_net, S01 => Data_18_net,
    S10 => Data_19_net, S11 => GND_1_net, Y => OPEN , FY =>
    Result);
end DEF_ARCH;

```

#### 4.1.25. tx\_ds\_gen.vhd

```
-- tx_ds_gen.vhd
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 68/81  
Filename: ASTCDetailedDesign

```
library ieee;
use ieee.std_logic_1164.all;

entity tx_ds_gen is
    port ( rst          : in      std_logic;
           d            : in      std_logic_vector (15 downto 0);
           decs         : in      std_logic;
           ck           : in      std_logic;
           ck40         : in      std_logic;
           dout         : out     std_logic;
           sout         : out     std_logic );
end tx_ds_gen;

architecture arc_tx_ds_gen of tx_ds_gen is

component clkint
    port ( a          : in      std_logic;
           y          : out     std_logic );
end component;

signal      a1          : std_logic;
signal      a2          : std_logic;
signal      a3          : std_logic;
signal      a3q         : std_logic;
signal      q0          : std_logic;
signal      q1          : std_logic;
signal      q2          : std_logic;
signal      q3          : std_logic;
signal      q           : std_logic_vector(15 downto 0);
signal      r           : std_logic_vector(13 downto 0);
signal      ckint       : std_logic;

begin

inst0: clkint port map (a=>ck40, y=>ckint);

process (ckint, rst)
begin
    if rst = '0' then
        q1 <= '0';
        q2 <= '0';
    elsif (ckint'event and ckint = '0') then
        q1 <= not ck;
        q2<= not q1;
    end if;
end process;

process (ckint, a1, rst)
begin
    if rst = '0' then
        q <= (others => '0');
    elsif (ckint'event and ckint = '1') then
        if (a1 = '1') then
            q <= d;
        else
            q<= '0' & q(15 downto 1);
        end if;
    end if;
end process;
```



# AMICA FOR AMS

## ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 69/81  
Filename: ASTCDetailedDesign

```
end process;

process (ckint, rst)
begin
  if rst = '0' then
    r <= (others => '0');
  elsif (ckint'event and ckint = '1') then
    r <= decs & r(13 downto 1);
  end if;
end process;

process (ckint, rst)
begin
  if rst = '0' then
    q0 <= '0';
    a3q <= '0';
    q3 <= '0';
  elsif (ckint'event and ckint = '1') then
    q0 <= q(0);
    a3q <= a3;
    q3 <= a2;
  end if;
end process;

a1 <= q1 and q2;
a2 <= ((q(0) xor q0) xnor q3) and (not a3q);
a3 <= r(0) and decs;
dout <= q0;
sout <= q3;

end arc_tx_ds_gen;
```



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 70/81  
Filename: *ASTCDetailedDesign*



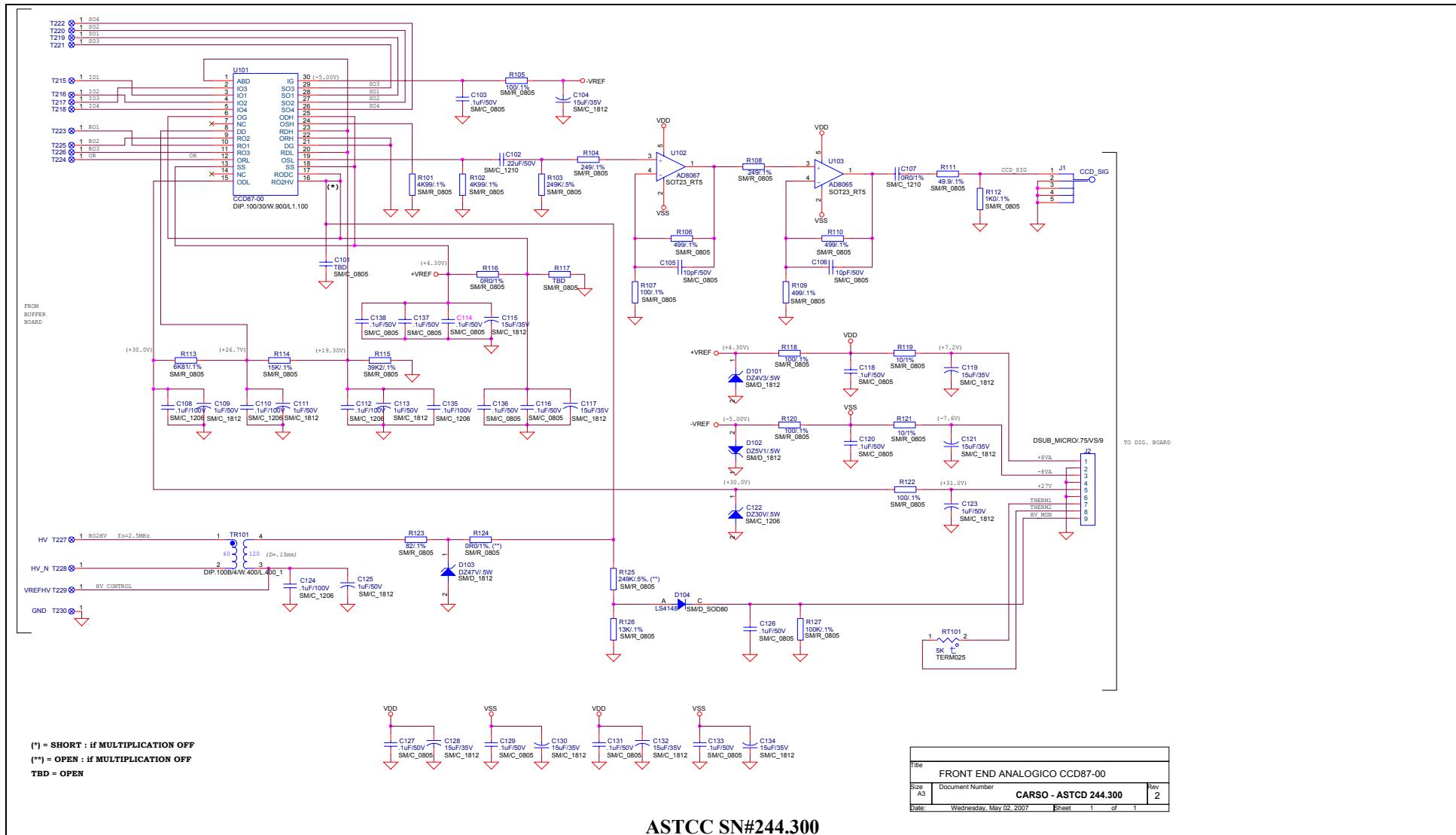
## AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 71/81  
Filename: *ASTCDetailedDesign*

## 5. SCHEMATICS

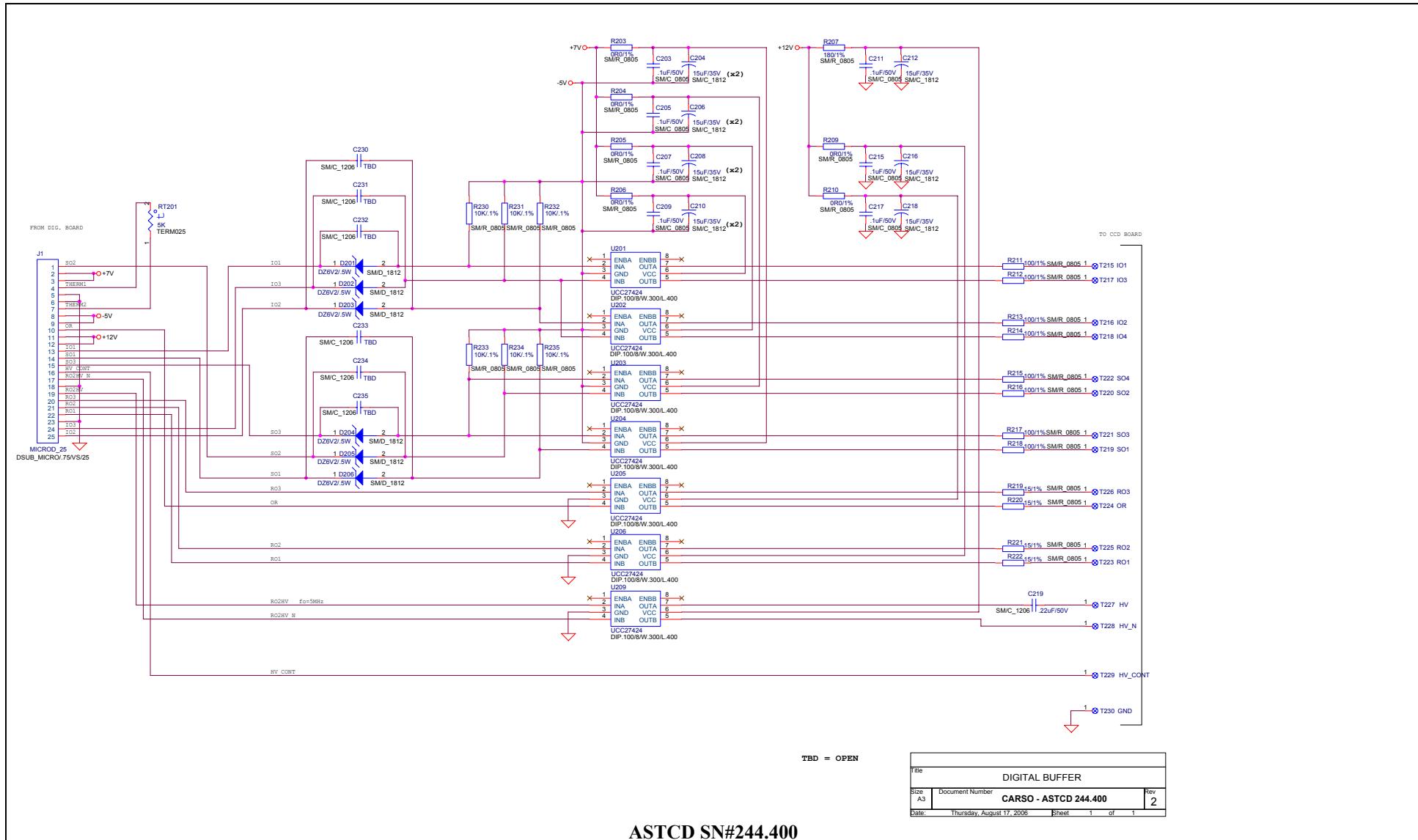
# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 72/81  
Filename: *ASTCDetailedDesign*



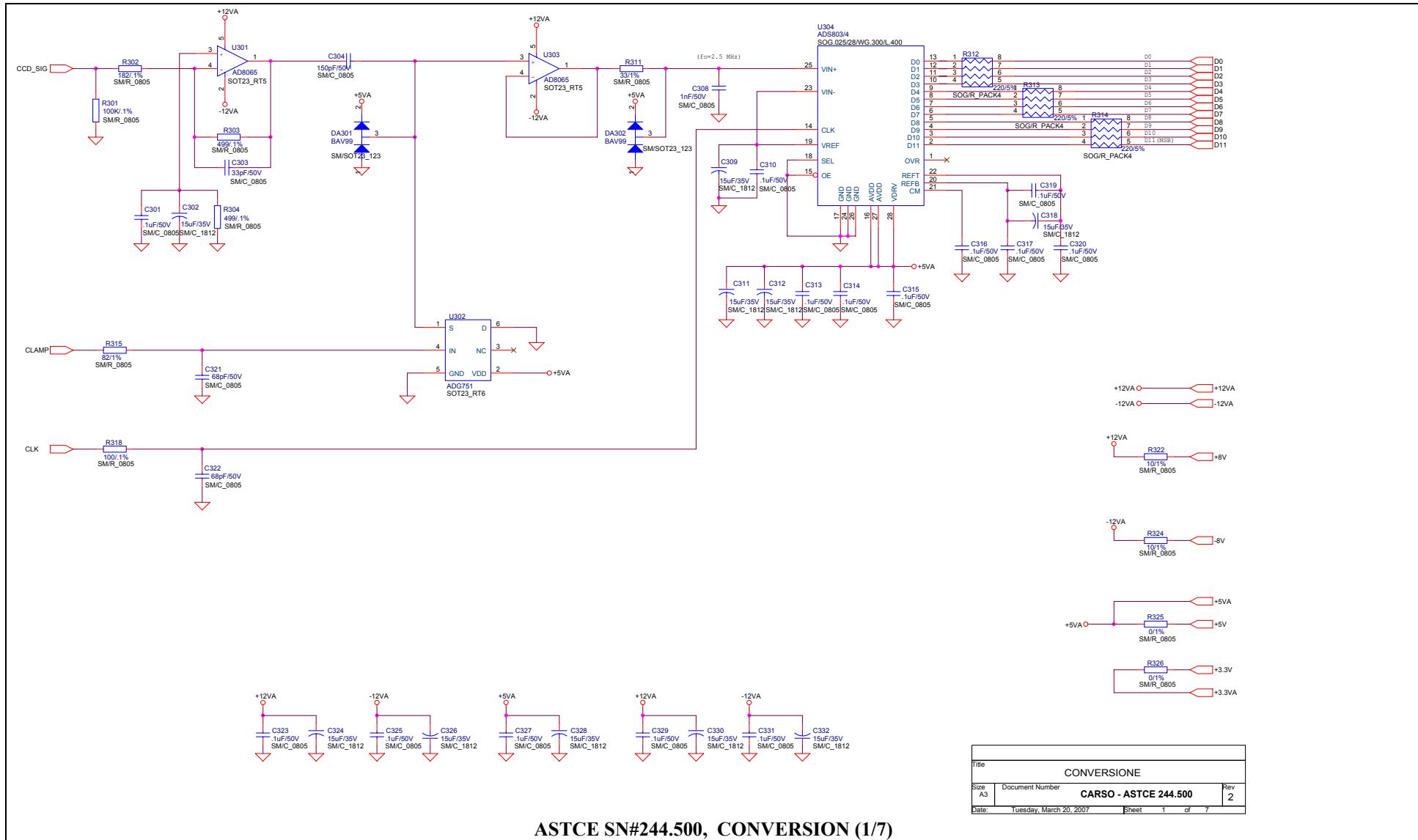
# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 73/81  
Filename: *ASTCDetailedDesign*



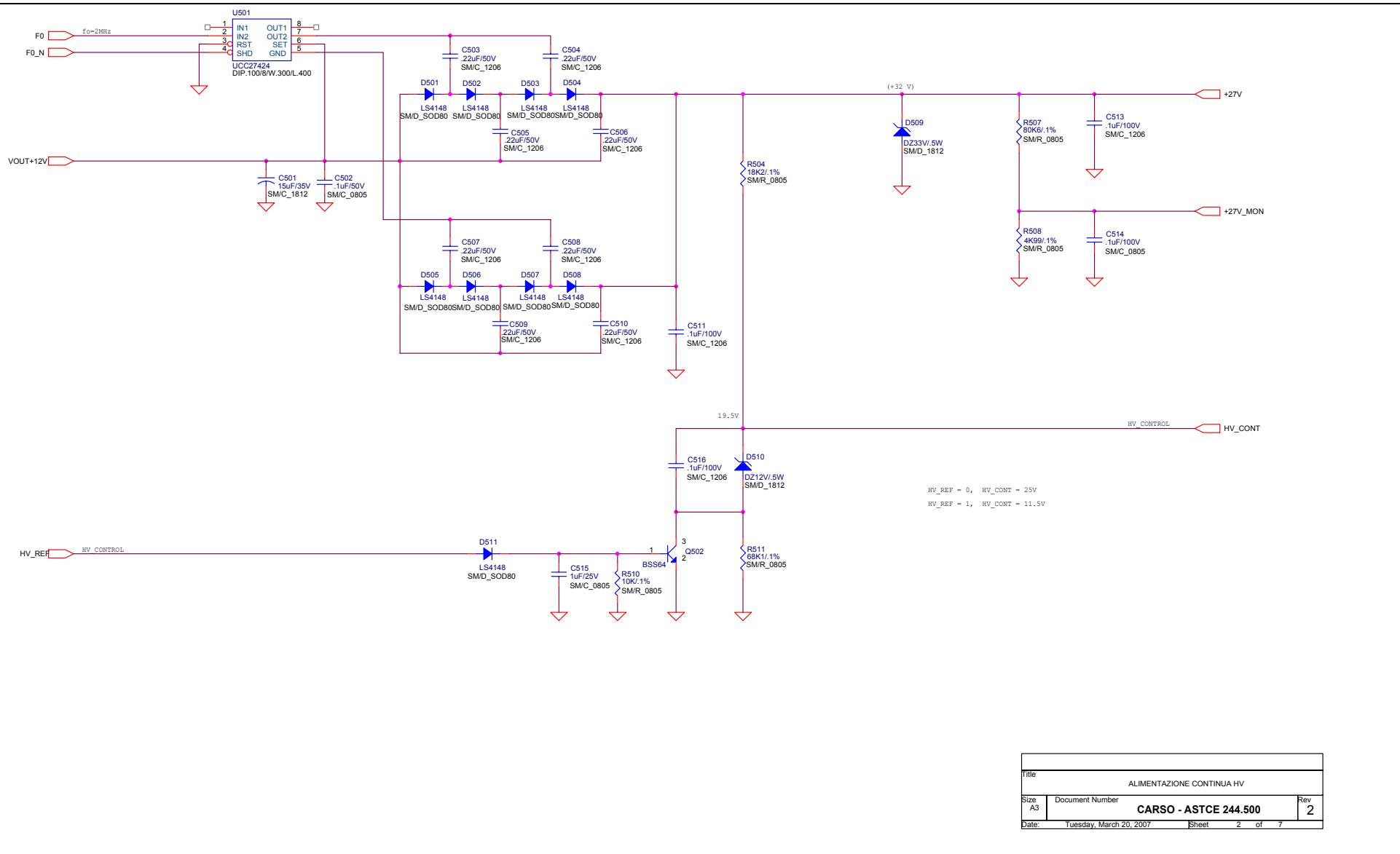
# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 74/81  
Filename: *ASTCDetailedDesign*



# AMICA FOR AMS ASTC Detailed Design

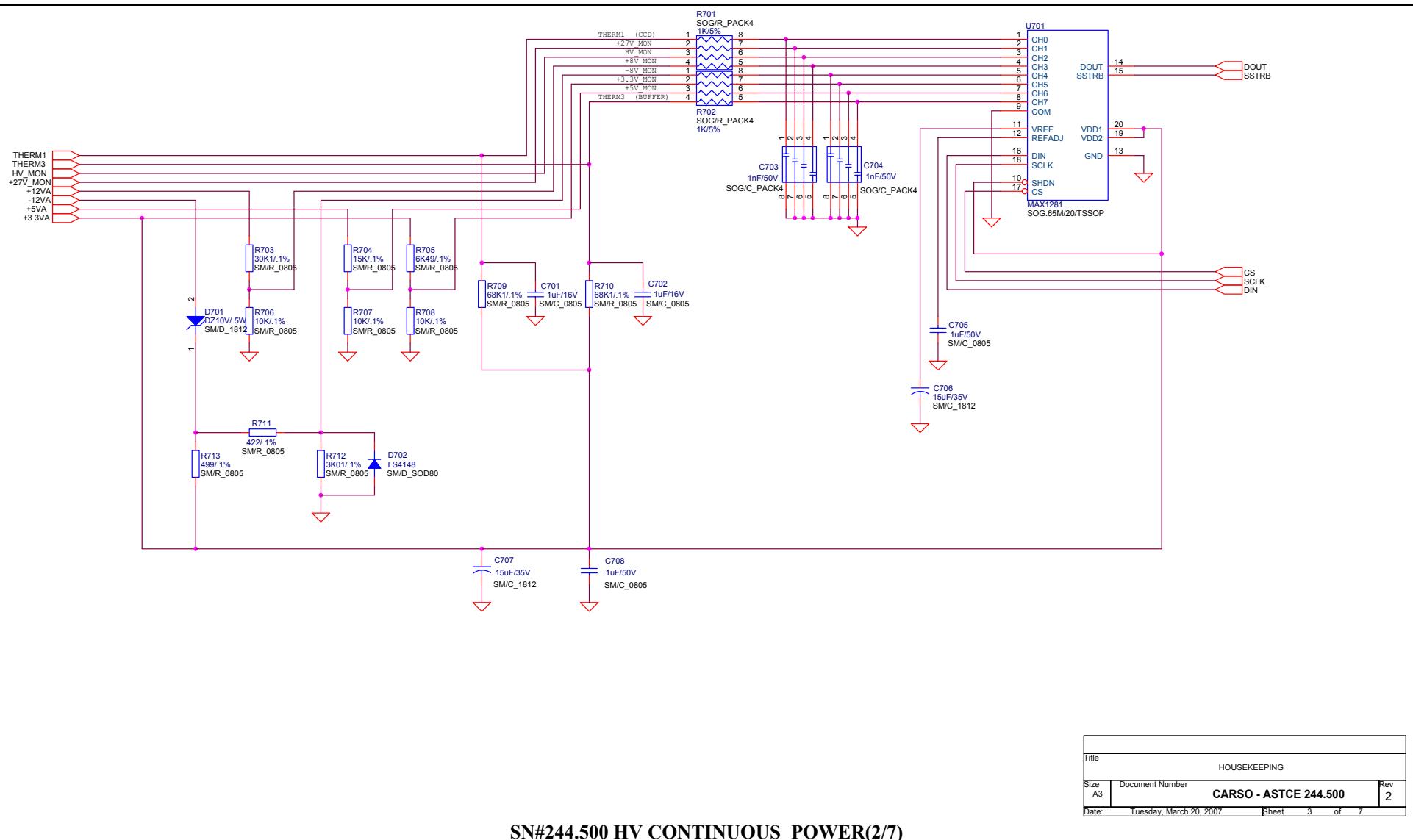
DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 75/81  
Filename: *ASTCDetailedDesign*



Title		
ALIMENTAZIONE CONTINUA HV		
Size	Document Number	Rev
A3	CARSO - ASTC 244.500	2

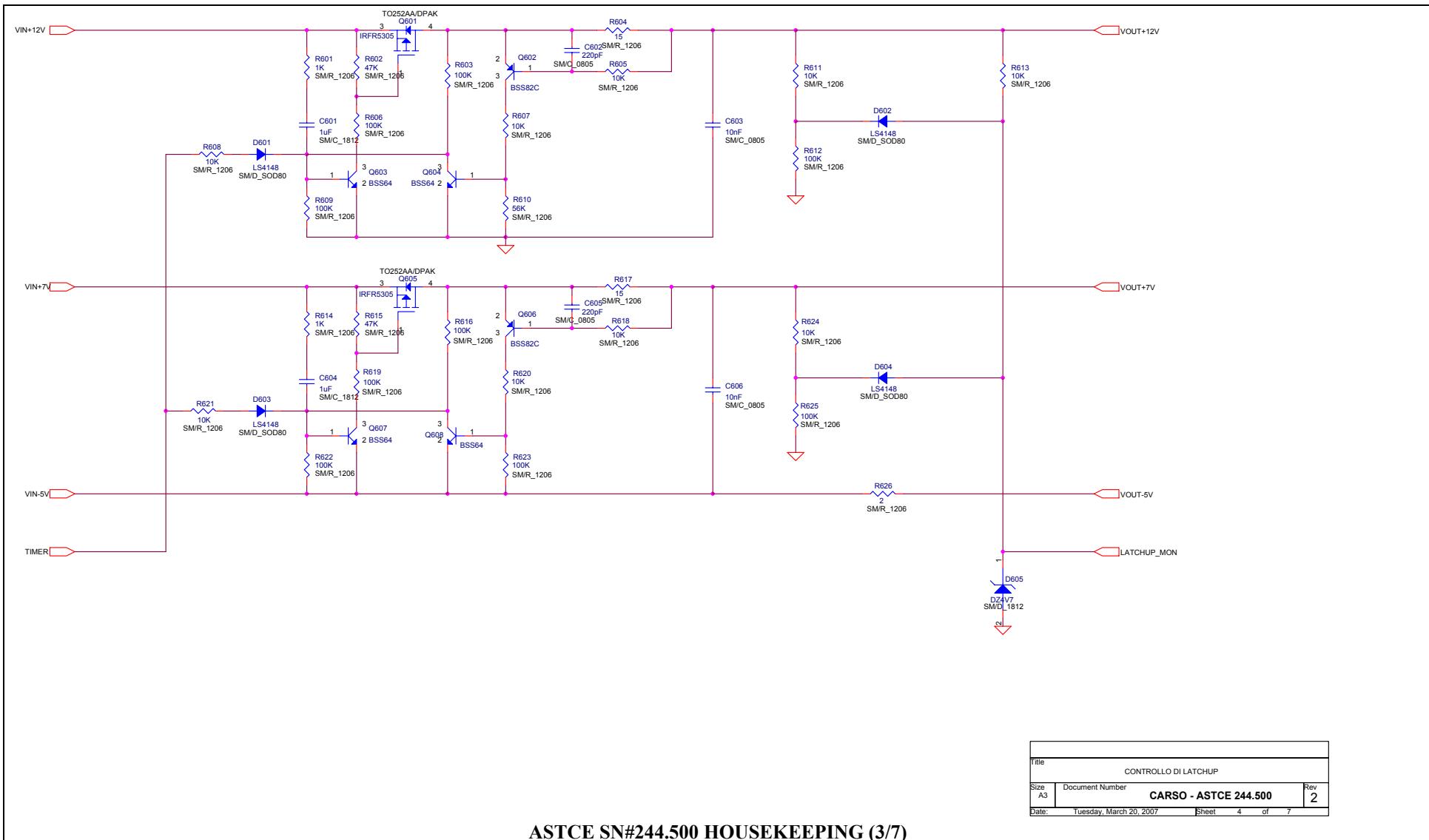
# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 76/81  
Filename: *ASTCDetailedDesign*



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 77/81  
Filename: *ASTCDetailedDesign*

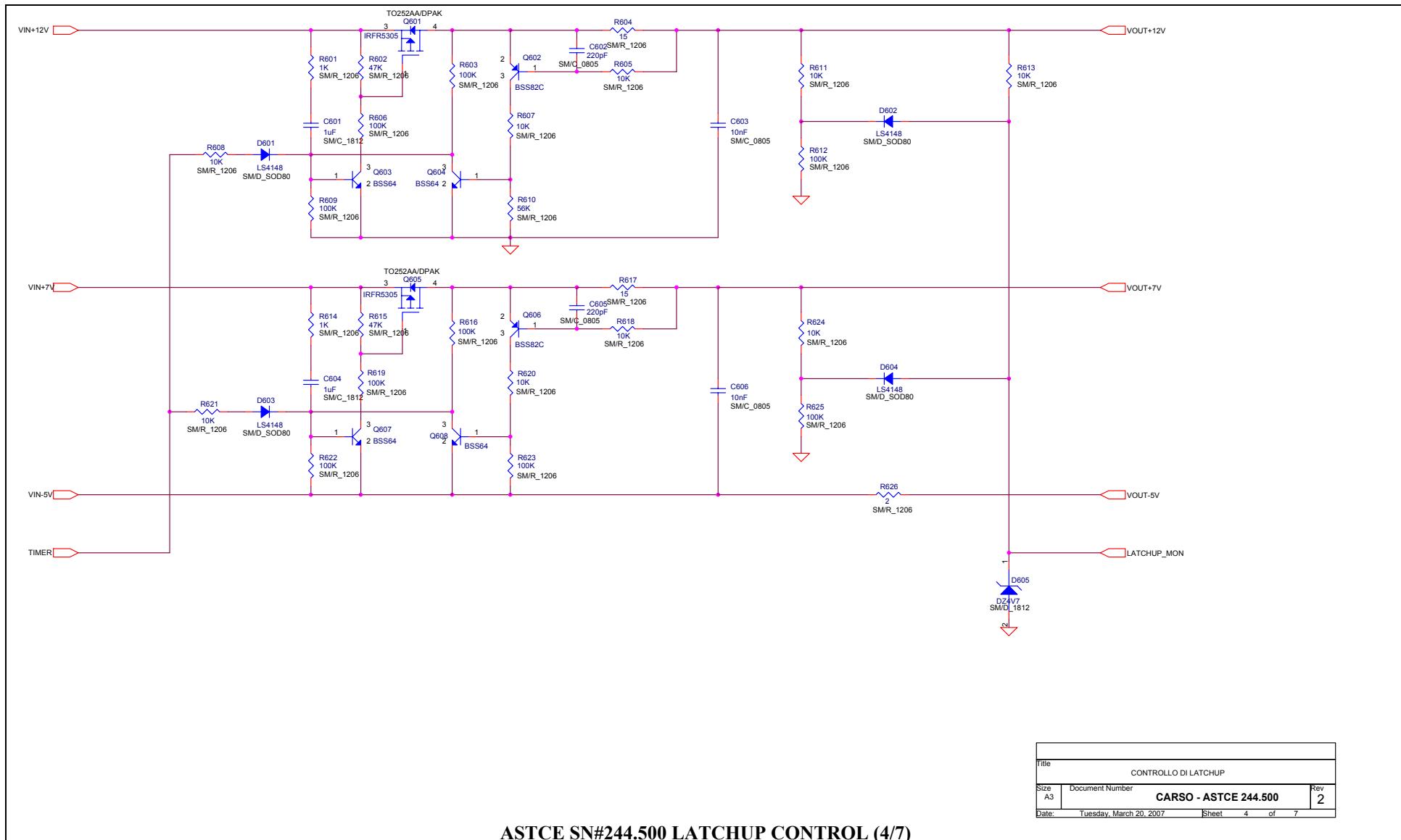


Title			
CONTROLLO DI LATCHUP			
Size	Document Number	Rev	
A3	CARSO - ASTCE 244.500	2	
Date:	Tuesday, March 20, 2007	Sheet	4 of 7

ASTCE SN#244.500 HOUSEKEEPING (3/7)

# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 78/81  
Filename: *ASTCDetailedDesign*



Title	
CONTROLLO DI LATCHUP	
Size	Document Number
A3	CARSO - ASTCE 244.500

Rev 2

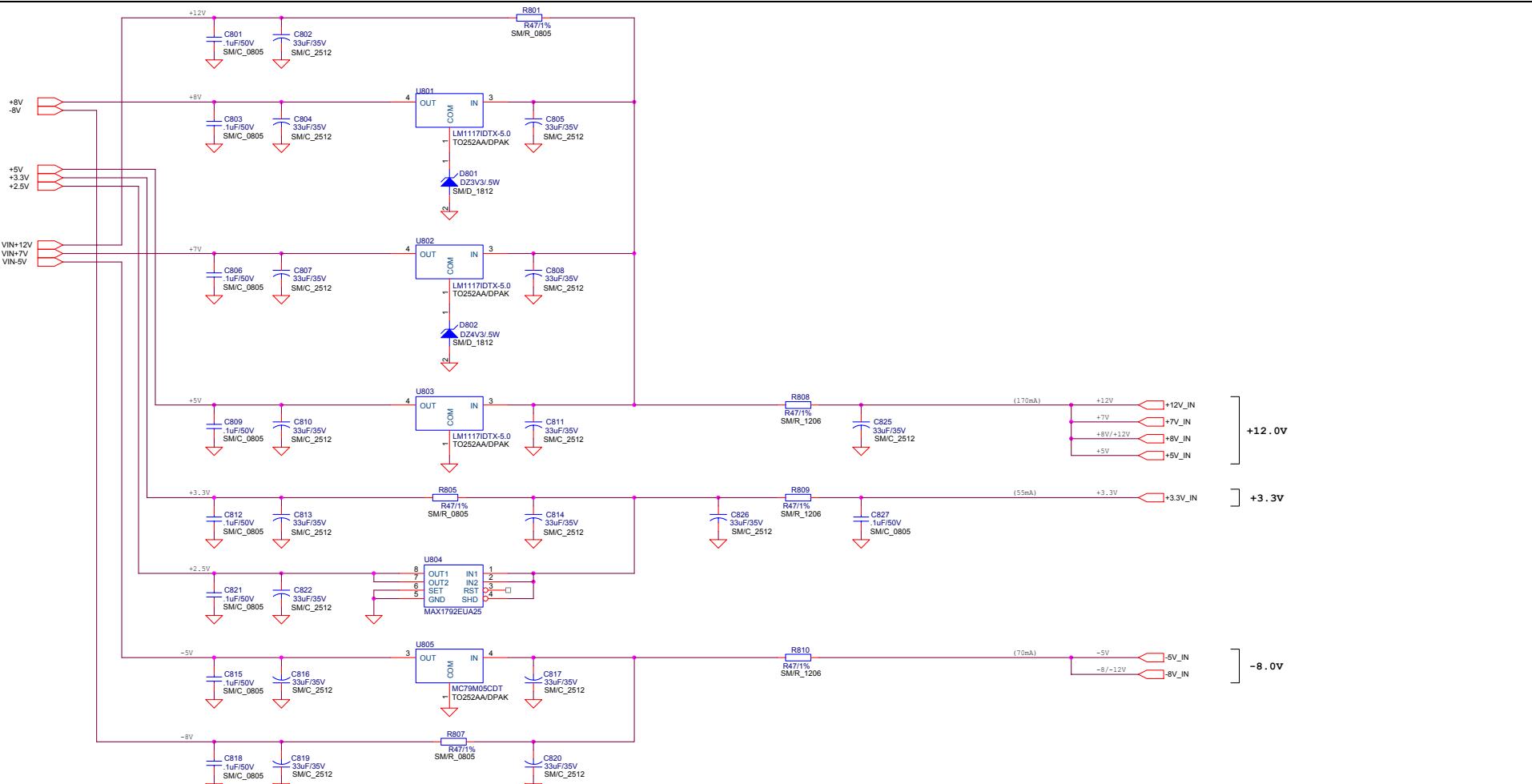
Date: Tuesday, March 20, 2007 Sheet 4 of 7

ASTCE SN#244.500 LATCHUP CONTROL (4/7)



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 79/81  
Filename: *ASTCDetailedDesign*



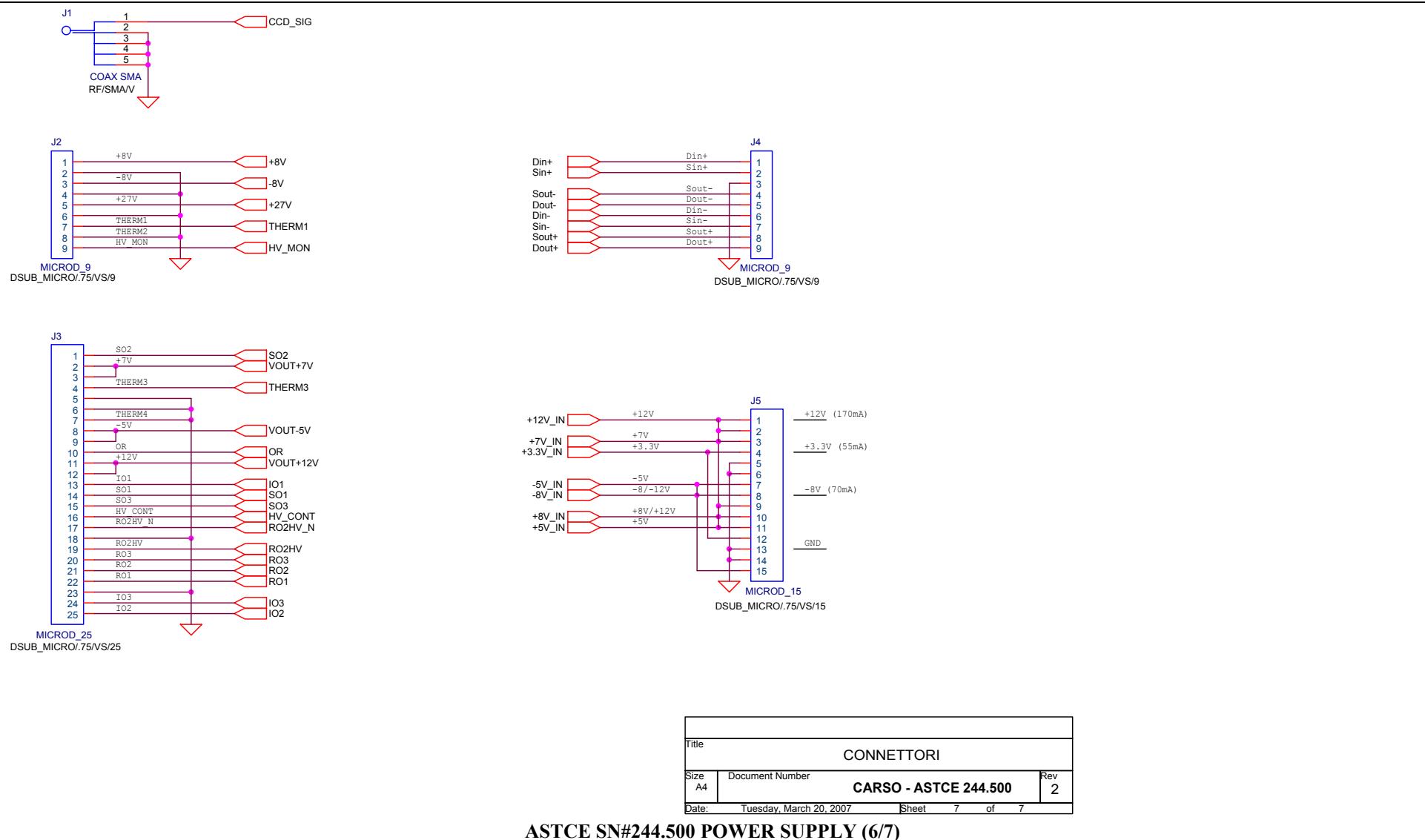
Title			
POWER SUPPLY			
Size	Document Number	Rev	
A3	CARSO - ASTCE 244.500	2	

ASTCE SN#244.500 DIGITAL CONTROLLER (5/7)



# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 80/81  
Filename: *ASTCDetailedDesign*





# AMICA FOR AMS ASTC Detailed Design

DOC.: AMST/ STCD /1/A  
ISSUE: 15  
DATE: Oct 06  
PAGE: 81/81  
Filename: *ASTCDetailedDesign*

